Palo Alto Research Center

A Guide to LSI Implementation

SECOND EDITION

By Robert W. Hon and Carlo H. Sequin

XEROX

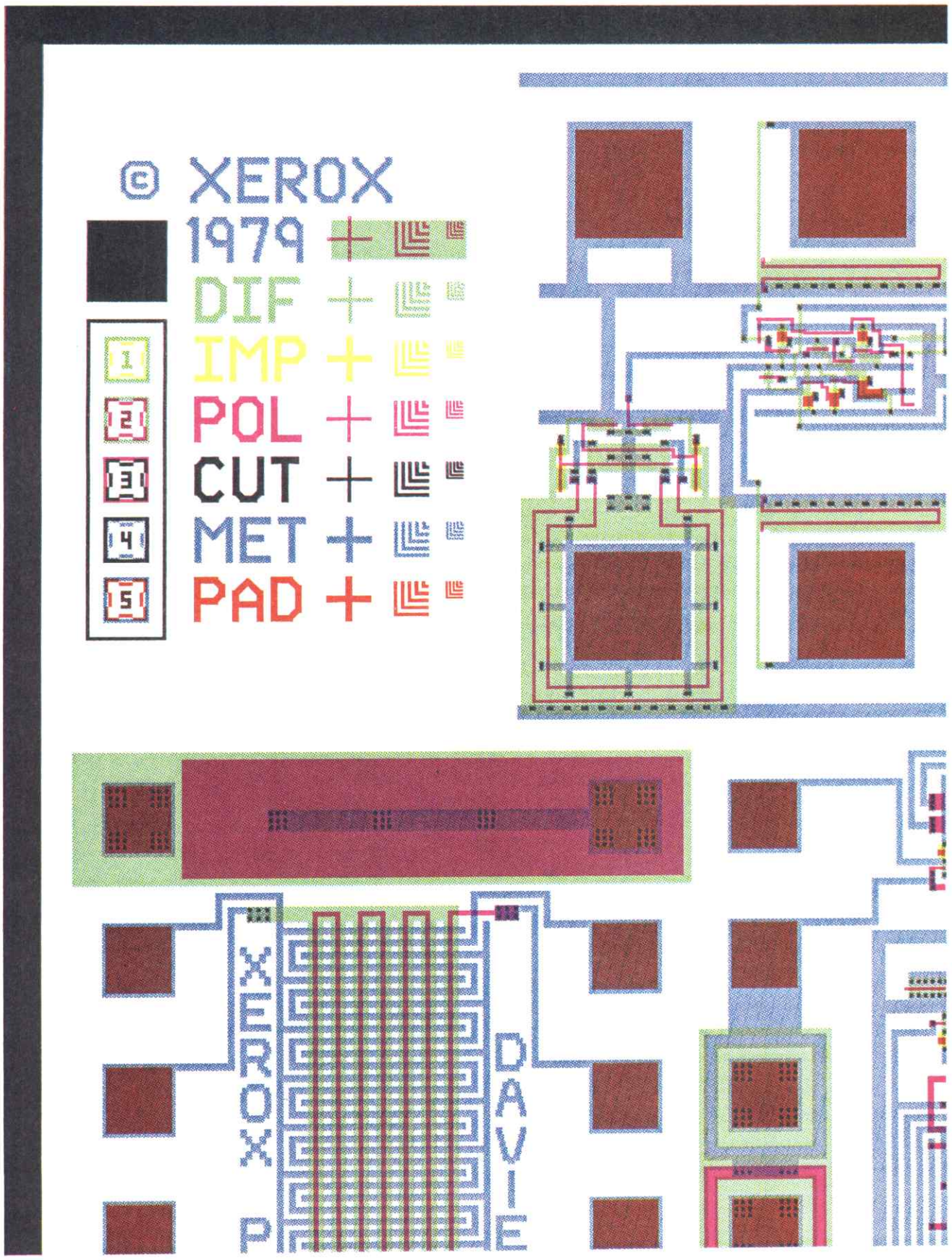
A Guide to LSI Implementation

Second Edition

Robert W. Hon and Carlo H. Sequin

SSL-79-7 January 1980

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304


© XEROX
1979
DIF
IMP
POL
CUT
MET
PAD
1
2
3
4
5
XEROX PARC DAVIE

to td,
from those who know

Foreword

During the past several years, the LSI Systems Area of Systems Science Laboratory (SSL) at Xerox Palo Alto Research Center (PARC) has conducted research in the architecture and design of integrated systems. One focus of that research, in collaboration with the Caltech Computer Science Department, has been the exploratory development of new design methodologies that simplify integrated system design.

Through this research, new design techniques have evolved and been debugged; these techniques can be more quickly acquired and more widely practiced by system designers than was possible in the past. It has been a period of discovery and iteration, moved forward by courses taught to university EE/CS students in which the students undertook LSI design projects as part of their class work. This process has lead to the publication of the textbook *Introduction to VLSI Systems* [Mead & Conway 1980] describing the new design techniques.

In order to make possible an "acid test" of the evolving design methodologies and of the resulting student designs, a parallel effort has been conducted in SSL to investigate, understand, simplify, and make more efficient the procedures for the *implementation* of LSI design projects. The term "LSI implementation" is defined here as the series of all tasks involved in going from a set of LSI design files to a set of packaged chips ready for functional testing. Implementation thus involves collecting and merging design files into starting frames, converting the merged files into patterning format, making masks, fabricating wafers, and packaging the resulting chips.

Bob Hon and Carlo Sequin have played leading roles in the SSL LSI implementation activities. They originated many of the new standards and procedures, and then validated these techniques while carrying out the implementation of several multiproject chip sets. These implementation activities have led to the discovery of new techniques that greatly reduce both the overall time for implementation and the cost of implementation per design project. Practical methods have been developed for simplifying the interfacing of design groups with mask making and wafer fabrication firms. A standard design-interchange format (CIF 2.0) has evolved from an early Caltech format and is now in widespread use in the universities and industry. All these and more are the subject of this report.

Bob and Carlo are to be congratulated on the success of these efforts and on the production of this timely and useful report. Only those who were near the action can visualize the imagination required and effort invested to reshape the "Silicon Valley folklore" concerning LSI implementation into a comprehensive, general, compact, and straightforward body of knowledge. Their work has been a key factor enabling the rapid spread of the participation of university students, faculty members, and researchers in the new field of VLSI system design.

Lynn Conway
18 January 1980

Preface

The 2nd Edition of *A Guide to LSI Implementation* is a compendium of information on the realization of LSI system designs. It is our hope that this report will enable a wider group of designers in universities and small systems firms to have their LSI chip designs implemented in an economical and timely way. This document also serves to establish some of the context for future SSL reports on research now underway concerning implementation systems for the remote-entry, fast-turnaround implementation of large numbers of VLSI designs.

The first edition of *A Guide to LSI Implementation* was hastily written in the summer of 1978 by a combination of PARC researchers, consultants and summer student employees. Among those contributing material were Wayne Wilner, Dick Lyon, and Rick Davies (PARC), Maureen Stone (Xerox-ASD), Bob Baldwin (MIT), Peter Dobrowolski (U.C. Berkeley) and Steve Trimberger (Caltech). Lynn Conway, Carver Mead and Doug Fairbairn each spent hours carefully reviewing drafts and offering suggestions. The eleventh hour efforts of all of these people allowed us to finish the first edition in time for Lynn's fall 1978 MIT VLSI design course.

In the year since that course we have had time to evaluate the strengths and especially the shortcomings of the first edition. In reorganizing and trimming it of irrelevant information, we hope to have improved the readability and utility of the work. New material has been added. The availability of electron-beam mask manufacturing facilities has made 5-day mask turnaround possible; we have included information to allow chip implementors to take advantage of this service. Largely through the efforts of Bob Sproull (CMU) and Dick Lyon (PARC), we have been able to address the many questions that have arisen about CIF 2.0. Chapter 7 is now a complete description of CIF 2.0 and serves as the official reference document. A new section has been contributed by MIT graduate student Jim Cherry sharing his experience from the successful 1978 MIT course. This edition has been further improved by input from SSL researchers Martin Newell and Alan Bell.

Xerox Corporation, Carnegie-Mellon University and the Advanced Research Projects Agency of the Department of Defense have been generous in their support of this work. Terri Doughty handled administration, editting, and much of the figure preparation. We are especially grateful to Dick Lyon, who helped us with sound advice and worked many hours solving the less than interesting problems of putting a report of this size together. He and Joe Maleson are responsible for the color plates. Finally, special thanks go to Lynn Conway for her enthusiastic support and encouragement of our work.

Bob Hon
Carlo Sequin

Palo Alto, California
18 January 1980

Table of Contents

CHAPTER 1

INTRODUCTION

Traditionally the design and development of integrated circuits (IC's) has been the domain of specialists with substantial training in this "art". With the emergence of powerful computer aids and of reasonably standardized IC processing techniques, IC design can be simplified to the point where it becomes a routine engineering step in the development of a special purpose system. MOS devices are particularly simple and straightforward as long as one stays away from the smallest geometries feasible. With reasonable, relaxed design rules the performance of standard MOS circuit blocks such as inverters, pass gates, buffers, NOR gates and composites of these blocks become as predictable as TTL circuits. Using a structured design approach, such as the one promoted by Mead and Conway in *Introduction to VLSI Systems* [Mead & Conway 1980], it is possible for people with only a minimal understanding of the device physics of a MOS transistor to produce operational integrated circuits of substantial size. Thus, a systems architect may now sit in front of an interactive graphics terminal and design the layouts of a set of masks for a special purpose integrated circuit. Such personalized IC's can greatly enhance the functionality of the system to be built or alternatively may dramatically reduce the total chip count for a system of given specifications.

In such a venture it is often not important to produce an integrated circuit of the highest layout density or of the highest performance, which could only be obtained by pushing the limits of present-day technology. Normally the main concern is to get a working chip with the shortest possible turnaround time. It is here that effective design tools and, even more importantly, the proper design methodology, are crucial. These issues are discussed in *Introduction to VLSI Systems.* The second, equally important part is to get the IC designs implemented. In an environment that is not already set up to routinely produce custom-designed IC's, the designer himself may have to be the driving force behind the implementation of the first few IC's. In this situation many months may be wasted because of unsuitable preparation or the unavailability of necessary information, leading to frustrating delays in the project schedules and to abandoning the custom-made IC approach altogether. These are problems that we hope this document will prevent.

Often several IC designs will be combined into a single multi-project chip. In this manner the cost of mask generation and wafer fabrication, as well as the organizational overhead involved in pushing the future IC through all critical stages can be shared among a larger group of people. In an academic or research environment this coordination of several experiments into one IC project is particularly important, so that not every student has to worry about all of the details of mask and wafer processing. Certain features such as test patterns for measuring device performance, alignment marks, and chip separation lines can be standardized and re-used in subsequent multi-project chips. Sticking to the same features, similar basic chip formats and established procedures

to generate the multi-project chips will help to streamline this process and enhance the chance for satisfactory results. Someone, therefore, will have to act as a coordinator. His or her first task will be to merge the different files describing the various IC designs with the starting frame containing the mentioned standard features. He will then interact with the mask house and the fabrication line, making sure that both places have all the information that they need and that there is no misunderstanding in what they are expected to do. In order to avoid unnecessary delays he should constantly keep track of the state of the project and try to effect smooth interactions between the various parties involved.

Converting an integrated circuit design into a finished, packaged, and tested chip is a time consuming task. Dozens of important details have to be observed to prevent disasters or costly delays. Up to this point a concise description of the specific details and necessary steps has not been available. Specific information had to be gathered from scattered sources including personal interviews with "old hands in the trade".

This document describes the real-life problems and details which the coordinator must be aware of to effectively carry out these tasks. But even the occasional designer of an individual IC should be aware of the overall process, so that he or she may better understand its impact on his own activity. As with any other systems implementation technology, the types of design aids and the methods of fabrication affect the type and quality of design which is done. The most effective systems designers will therefore understand at least the basic aspects of design aids and checking tools as well as mask and wafer fabrication.

Throughout the text *italics* are used to introduce vocabulary that the designer should know. No attempt is made to tabulate precise definitions of terms, but enough information can be inferred from the context that the reader can search for more details if necessary. Chapters 2 through 5 outline the basic path from IC design to the finished product. Chapters 2, 3 and 5 should be studied even by people who never dream of becoming coordinators of multi-project chips, since they set the stage for proper IC design. For the coordinator, Chapter 4 is vital. Chapter 6 gives an example of a multi-project chip produced at Xerox PARC during the summer of 1978. It carried 10 experiments including a wide range of logic, arithmetic and memory circuits and a test pattern. Chapter 7 provides description of the Caltech Intermediate Format. It serves as the syntactic and semantic definition of CIF 2.0 and will be updated as necessary. Appendix A contains a listing of the instructions sent to the mask house concerning this particular chip. Appendix B is a listing of some integrated circuit manufacturers who can provide valuable information and insight into most phases of IC implementation. It is by no means exhaustive, and the reader should not hesitate to make his own contacts where possible. Appendix E provides pointers to articles and texts covering particular aspects of IC implementation.

We cannot over-emphasize the importance of actually participating in the process of implementing chips. This document and a willingness on the part of the project coordinator to ask questions should be sufficient for turning IC design files into silicon. While there are many

difficulties to be overcome, the task can be managed well, even by those with no experience in IC implementation.

References

[Mead & Conway 1980]
C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, 1980.

CHAPTER 2

IC DESIGN TOOLS

The key to fast-turnaround integrated circuit design is the emergence of powerful tools that make it possible for the designer to be effectively assisted by computers. For example, the availability of interactive graphics terminals allows him to enter ideas into a design system quickly. Layout languages are being developed that permit the entry of designs at a high level of abstraction. Computer generated displays and plots of the mask levels of an IC are important to close the man-machine loop and are indispensable in the final debugging phase. There are additional ways that the computer can assist the designer. A "mechanical" check for design rule violations helps eliminate potential problem spots in an IC design. Circuit simulators can be used to predict the performance of critical parts of the design and logic simulators can be useful in testing the correctness and timing of the overall chip.

In all these interactions with the computer, there is an underlying database that contains some description of the integrated circuit. This may be a circuit description, a layout topology description, or a description of the geometry of the masks required to fabricate the IC. The database may be more or less sophisticated, and designers can interact with it at different levels. At the lowest level it simply stores data in some internal format, with no "intelligent" processing. In this case the designer may have to enter his design as a low-level geometrical description, and the only thing that can be done with the information stored is display it on various output devices. At the other extreme, very sophisticated design tools may one day accept a high-level description of a particular system and compile a complete layout. Some of the IC designer's principal tasks are:

- Entering design geometry
- Outputting design geometry (plotting, printing)
- Documenting the design
- Checking for design rule violations
- Checking for logical errors
- Simulating the behavior of a design
- Testing the actual integrated circuit.

This chapter provides an overview over the types of tools available and under development for these various tasks, starting with the two elementary operations: inputting and outputting mask geometries. It should be pointed out that an ASCII keyboard and a lineprinter attached to a general purpose time sharing computer can support a set of design tools that has proven adequate for IC designs of substantial size. In general it is worthwhile to expend significant effort to develop your design tools. Even for an investment of several weeks, the return on your time will be quite

high if you intend to do more than just one small design. Our experience has shown that in generating one's first designs it is not unreasonable to spend half of the time in enhancing (or creating) software tools.

2.1 Entering Your Design

The lowest level for entering the description of an IC is a point by point description of the geometry on each mask level. In the absence of any sophisticated design aids a detailed drawing of the design can be done with colored pencils on graph paper. From there the coordinates of the corners of all polygonal shapes can be read off and typed into a program that constructs some internal representation of the design for later plotting. A *digitizing table* is a means to input the coordinates of vertices directly from a drawing without the need for the explicit typing of numbers. A pointing device such as a stylus or a crosshair is placed on the crucial points in the drawing and the position on the digitizing table is automatically converted to a corresponding set of coordinates.

A *low-level* description of mask geometries that is used by IC designers at several universities is the Caltech Intermediate Form (CIF), described in Chapter 7. CIF has simple language constructs to specify elementary shapes such as rectangles, polygons, flashes, and wires. Since CIF is intended to be machine-generated, rather than human-generated (although it can and has been successfully used by designers without better tools), it is rather tedious to use the language to enter designs by hand. One alternative to entering CIF directly is to write a short program to provide an interactive input environment similar to a text editor. Such an input program could prompt for the next entry, provide suitable default values, warn of format errors, permit convenient iteration of cells, provide user selectable reference points and relative coordinates, and handle the management of the generated files. An example of an input interface, written in the language Pascal, has been developed at the University of California at Berkeley [Krause 1979].

A more efficient input device for layout geometries is an *interactive graphics editor*, engineered for the special needs of IC layout. The necessary hardware includes a pointing device, used in drawing the layout, and a graphics display to show the design as it is being entered. Ideally the system should be as easy to use as a pencil and paper when producing a first rough sketch, yet it can contain features to help the designer avoid mistakes or be more productive. For instance, it may generate layouts on a specified grid, with all edges by default parallel to the coordinate axes and all interconnection paths of a presettable default width (for example the Icarus system [Fairbairn & Rowson 1978]). Another approach is to allow the designer to work on a loose grid [Williams 1977]. This facilitates the easy repositioning of items, since only relative positions are indicated on the screen. Once the layout is completed, a program fills out features to default dimensions and compacts the geometric layout into a final mask specification. In either approach it should be easy to define parts of the display as a named cell that can be called at a later time to be inserted into different places in the design – if necessary, with transformations (rotation, scaling,

mirroring) applied.

2.2 Hardcopy Output

Even when a system with interactive graphics terminals is available, the ability to produce hardcopy checkplots is essential for documentation and error checking. Hardcopy output allows the designer to paste together an overall view of a complicated design with enough resolution for error checking, marking design changes, and inserting comments. Good checkplots must show several mask levels simultaneously in such a manner that it is clear exactly which levels contain an active feature at any specific point. Such checkplots are most easily read if all mask levels give an impression of being semi-transparent. Filled-in color features have proven very effective and easy to read, however, color output devices are still quite expensive.

Inexpensive checkplots of low resolution can be obtained from an ordinary line printer using different characters to represent different layers and separate or overstruck characters to show overlapping layers [Gibson & Nance 1976, Larsen 1978]. A few hundred lines of code are sufficient to produce such a plot from any CIF file.

Color line-drawing plotters provide output of much higher information density. While a drawing of the polygonal outlines alone may be relatively hard to read, the readability can be improved if internal areas of the shapes on each layer are crosshatched in the appropriate color. Unfortunately this results in very long plotting times. Furthermore, intersecting shapes within the same layer, which during mask generation get merged into a single polygonal shape, may show all the internal boundaries as well, which makes these outline drawings rather confusing.

Electrostatic dot raster plotters (such as those offered by Versatec or Gould) generate high resolution, filled-in output in a matter of minutes. The various mask levels can be distinguished by different gray-pattern shading or by different stipple patterns. These stipple patterns should be properly selected so that any individual mask can be seen in the presence of any arbitrary combination of other mask levels. The outline of the structure of such a plotting program is presented in Section 7.3.

2.3 High-Level Descriptions

So far we have only dealt with the design at its lowest level description, the geometry of each mask level. In many instances the efficiency of the designer could be significantly enhanced if he or she could communicate with the machine at a higher level. A first step in that direction is to use a hierarchical organization in the design, subdividing the overall problem into more easily handled subproblems. In IC design the elements of the hierarchy are called *cells* or *symbols*. An *instance* or "use" of a cell can stand alone or be embedded as a subpart of another cell. This means that the

contents of the cell are inserted into the design at certain points in much the same way that procedure calls are embedded within other procedures in an ordinary programming language. Such a hierarchical approach reduces the amount of information that has to be handled and stored explicitly, and is the key to modular debugging.

The next step is to provide a library of such cells with suitable parameters, for example a general adder cell that is called with a parameter specifying the width of the data path, or a PLA that takes as an input the array specifying the positions of all connection points in the two NOR planes. In the same spirit a set of generic cells could be called in conjunction with a *technology specification file*, which then returns a cell with the proper linewidths and registration tolerances for that particular technology.

Rather than specifying a design through geometrical shapes or parameterized layout cells, it would be preferable to specify the designers intent at an even higher level, for example by the equations for a block of boolean logic or by the state diagram for a finite state machine. For this purpose a description language and a compiler that produces a corresponding layout are required. In particular the PLA mentioned above could be called by the set of its logic functions, and the compiler would take care of the minimization and call the proper number and type of internal and peripheral cells to make up the PLA.

This brings up the point that for any design there exist a number of different *representations*: register diagram, circuit, layout topology, mask geometry, behavioral description or descriptive text. An advanced design system might contain the information for all or some subset of relevant representations in an integrated data base, so that the designer can view his design in a number of ways. Ideally this data base would automatically reflect changes to one representation in all the others.

A somewhat less ambitious system may have one master design representation that reflects the intent of the designer and in which modifications can be introduced. Other representations are then slaved to this master and could be automatically updated, interactively or by a separate compilation phase. A feasible design system of the near future may permit the designer to specify his designs at the register transfer level in a language like ISP [Bell & Newell 1971]. From that description general, parameterized cells, possibly represented by their topology in stick diagram form, are called from a library and fleshed out to the proper dimensions based on a design rule or technology file. The resulting blocks may be further adjusted in size for the best fit [Johannsen 1979] and then be submitted to an automatic routing program that wires the blocks together. After that, actual mask geometries can be generated, from which the values of parasitic circuit elements can be calculated. These values together with connectivity information stored in the data base form the input for a simulation program. To check the completed design, the output from this simulation is compared to the behavioral description of the overall design at the register transfer level.

The next higher level of input specification should include features such as user-definable subroutines, parameterization of objects and conditional branching. With such features the layout

language has the power of a general purpose programming language plus all the specifically built-in features that make it suitable for layout purposes. In particular, the parameterization of objects permits one to base the position, size or shape of a geometrical feature on the position or size of some other feature. Thus, for example, objects could then be moved around while signal paths or wires remain connected to the proper points. Furthermore, the position of interconnection points could adjust themselves to match the corresponding locations in adjacent cells, and power supply lines could automatically be widened so that they can handle the total current required by the connected cells. Layout languages can, in principle, be implemented in any computer language (see the description of ICLIC in [Stone 1978]).

In conclusion, the complexity of VLSI designs forces us to develop the necessary tools to communicate with the machine at a much higher level than was traditionaly possible. Automatic compilation of various representations will dramatically reduce the occurrence of errors and keep design time within acceptable limits.

2.4 Design Rule Checking

[contributed by Wayne Wilner, Xerox PARC]

Where automated design systems are not available and the mask geometries are generated by a human designer using low-level tools, the chance for errors and design rule violations is high. Thus, careful checking of the layout is mandatory. In the absence of any checking programs, the designer checks his layout by inspecting the hardcopy output plots. While this may sound like an impossible task for designs with more than a few hundred features, there are a few things that come to the aid of the designer. Humans possess tremendous pattern recognition power. Each mask level of a modular, densely packed IC has a surprising amount of structure and regularity, and design rule violations such as lines that are too narrow or too close together stand out clearly. Of course, any intentionally introduced regularity, such as the use of arrays of the same cell, also facilitate checking. It may be astonishing, but even people who do not understand the details of a particular design can readily detect certain violations on a checkplot. To check rules that concern more than one level, it is advisable to jointly plot the levels involved for easy inspection. Combinations of particular importance are poly-diffusion-cut and metal-cut. In general every design should be carefully inspected by at least one other impartial designer.

The mechanical nature of design rules allows some checking to be performed by computer. Starting from a CIF file, the various shapes in the same mask level that touch or overlap are merged. Subsequently the positions of suitably oriented edge vectors of all polygonal shapes can be compared to one another to check for adequate distances. For large layouts a suitable ordering and grouping of all features is necessary to prevent the compute time from increasing with the square of the number of features.

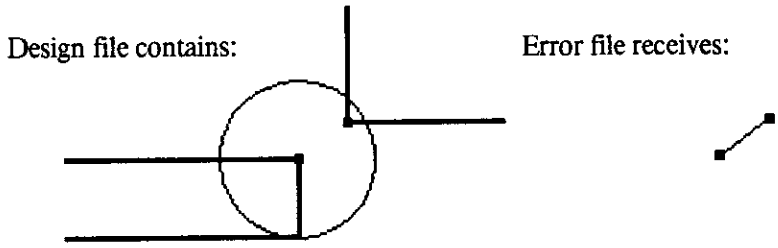
Error-checking programs embody the rules for a particular process and examine CIF files or pattern generation tapes for violations, reporting each instance in terms of coordinates or patterns, along with the nature of the violation. Design rules typically assign minimum distances to:

dimensions of features, such as breadth of runs or size of contact cuts;

spacing between features in the same layer, such as distance between runs;

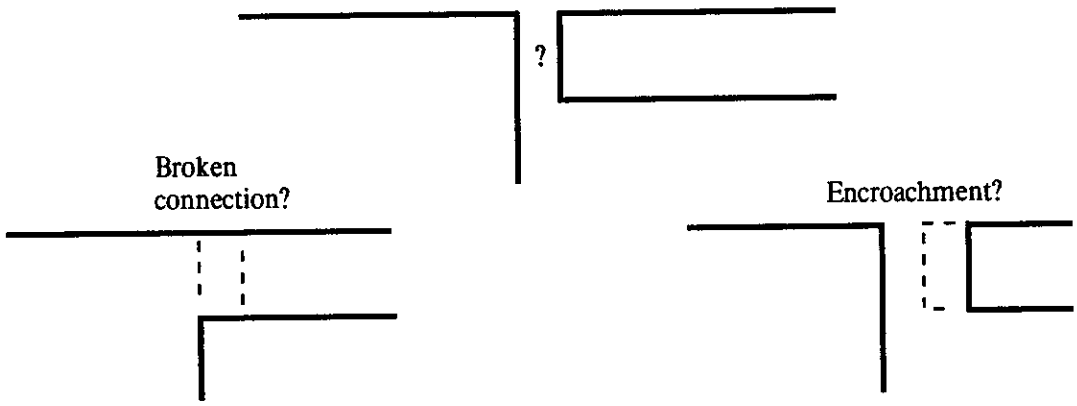
spacing between features in different layers, such as overlap of metal and contact windows.

For example, suppose unconnected areas of polysilicon must be 2λ apart. In the diagram below, a circle of radius 2λ centered at the upper right corner of the left-hand area reveals that the right-hand area is too close.



This design rule violation may be reported in terms of a line segment, that is, two points, one at the periphery of each area, and their (insufficient) separation. It is a non-trivial problem to present violations to the designer in the most convenient way. The output from such a program could simply be a list of the positions and levels of all pairs of vectors violating a particular design rule. Alternatively an extra error "mask" level, which can be superimposed on the regular layout plots to indicate the location and type of violation, may be easier to read.

An inherent limitation of design rules comes from their pertaining solely to the lowest level of detail. Consider the following diagram. Two areas are separated by less than their minimum spacing.



It is clearly a design rule violation, but is it a broken connection or is it an encroachment? The designer will have to decide and fix it appropriately.

In laboratories that experiment with different processes, the critical distances may vary from month to month. In experiments with custom circuits, the objective may be to find how exceptions to the design rules can be exploited. Therefore, while the types of rules may be rigidly bound into a checking program, specific distances should be parameterized. The difficulty with such an approach is that design rules used in an industrial environment are often rather complicated, involving conditions such as: "polysilicon must extend at least four microns beyond diffusion, unless the gate dimensions are small, where six microns are required". It is an unsolved problem to mechanically create a program that can efficiently verify geometrical constraints of varying nature. For the much simpler design rules used throughout *Introduction to VLSI Systems* [Mead & Conway 1980] the problem is tractable.

2.5 Checking for Other Errors

Many fatal errors in a design do not exhibit themselves as violations of design rules. Consider logic errors, state machines that are initialized to terminal states, or transistors that are wired incorrectly, but within the given design rules. Consider an array of cells that are supposed to abut exactly, but are either separated or overlapping; if the space between them is larger than the minimum spacing for all layers, design rules may be observed while the array is grossly in error. Consider the placement and continuity of busses. These errors are representative of flaws for which the designer is singularly responsible.

Many such errors can be spotted on relevant subsets of layers. A plot of metal and contacts can reveal errors in continuity which would otherwise be lost in the details of a full plot. Alternatively, a plot of poly, diffusion, contacts and implants enables one to check their important overlaps. This technique is very effective if the plots are large, clean, and of high contrast.

Hardest to find are connections that are almost correct such as a connection to the Q-bar output instead the Q output of a flipflop, or a connection to the wrong bit in a wide bus. In particular, wide-ranging interconnections between cells are error prone since they are easily lost in the jumble of a large checkplot. Errors of that kind must be detected by logical checks or simulation. In manual debugging, such errors are often found if the actual signal path is traced with a colored pencil on your hardcopy output.

Automatic compilation of a layout from previously defined and debugged cells will drastically reduce the number of errors. But even with such a structured approach, rather basic errors may occur if adjacent cells are either improperly placed or if the individual cells were not specifically designed to tolerate arbitrary neighbors without creating a design rule violation. In manual debugging, cell and array boundaries and all interconnection points should be checked with

particular care.

2.6 Simulation as an IC Design Tool

[contributed by Richard Lyon, Xerox PARC]

Simulation is a design technique widely used in a variety of engineering disciplines. When it is too difficult to verify the correctness of a design by inspection, by proof, or by test, simulation may help. Simulation allows the designer to test a design before building it, by modelling in detail the components from which the design is built, and by computing their interactions under various conditions. Simulation is useful at many levels in integrated circuit and system design; system-level, register-transfer-level, logic-level, and circuit-level simulators are useful at various stages of the IC design process. A related activity is the design of IC fabrication processes, which can benefit from process simulation; the simulation of process variations may become more important as VLSI approaches the physical limits of device sizes, where the set of devices used by the system designer must be carefully matched to the technology.

Unfortunately, not many generally useful simulators are readily available. Even when such a program is available to run on your computer, the problem of preparing data in a form suitable to the simulator can be formidable. It is easy to write a register-transfer-level simulator, for example, but the part that would make it useful, an automatic link from the design language to the simulator input language, is much harder to implement. The lack of commonality of design methods in the digital system design field have resulted in a delay in the availability of such programs. In the circuit design field, on the other hand, the method of design has traditionally been standardized to drawing by hand on paper the interconnection of standard types of lumped circuit elements. From here it is logical to assume hand translation to the language of a circuit simulator. For this reason, circuit simulators are widely available in standard languages (Fortran IV). Two such simulators, somewhat tailored for IC simulation, are SPICE from U. C. Berkeley, and MSINC from Stanford. Their input languages are similar, and one example should serve to illustrate both.

As an example, we have simulated the output pad driver called *PadOut*, which was designed in *Icarus* (Integrated Circuit ARtwork Utility System, an interactive layout design system) according to the Mead and Conway design rules, with lambda equal to 3 microns. This is a driver intended to interface NMOS chips to other popular logic families, at speeds and voltages comparable to TTL. It uses push-pull enhancement-mode output drivers, driven in turn by *super-buffers* (see [Mead & Conway 1980] chapter 1). The fanouts are generally somewhat higher than the theoretical optimum of e, to reduce space and power at the expense of speed. Figure 2.6.1 is the Icarus layout picture of PadOut; notice that the output transistors are both wrapped around the pad. The schematic diagram is shown in Figure 2.6.2; it includes node numbers and element names which are needed for translation to the simulator input language.

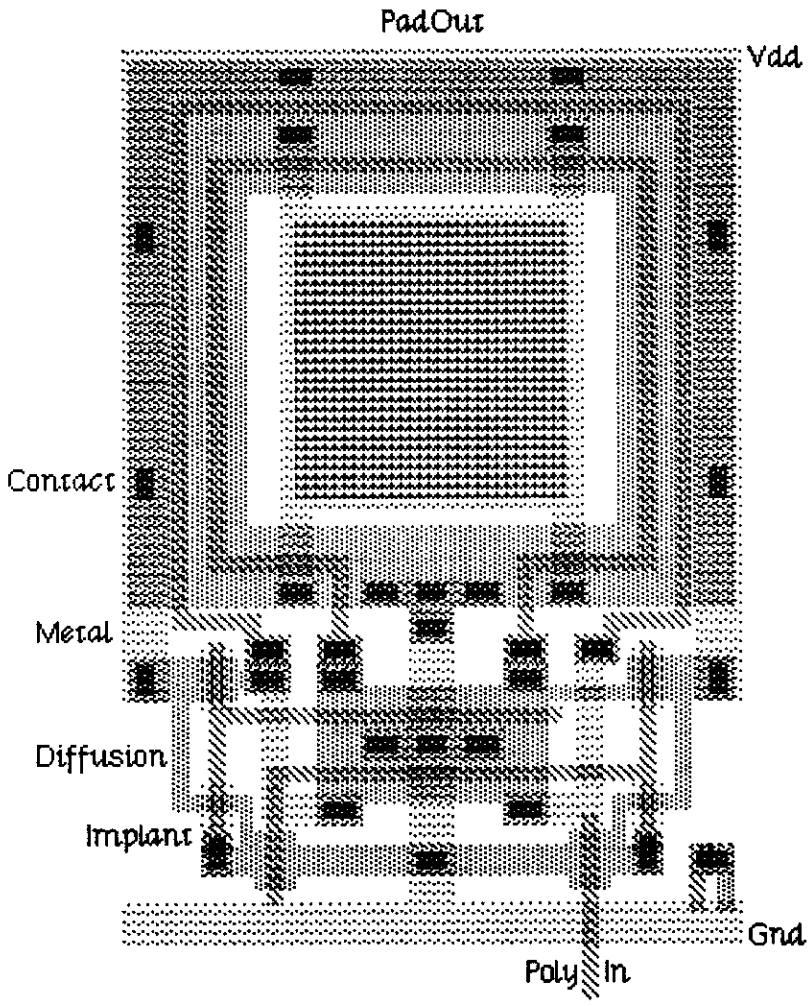


Figure 2.6.1. Icarus Layout of "PadOut"

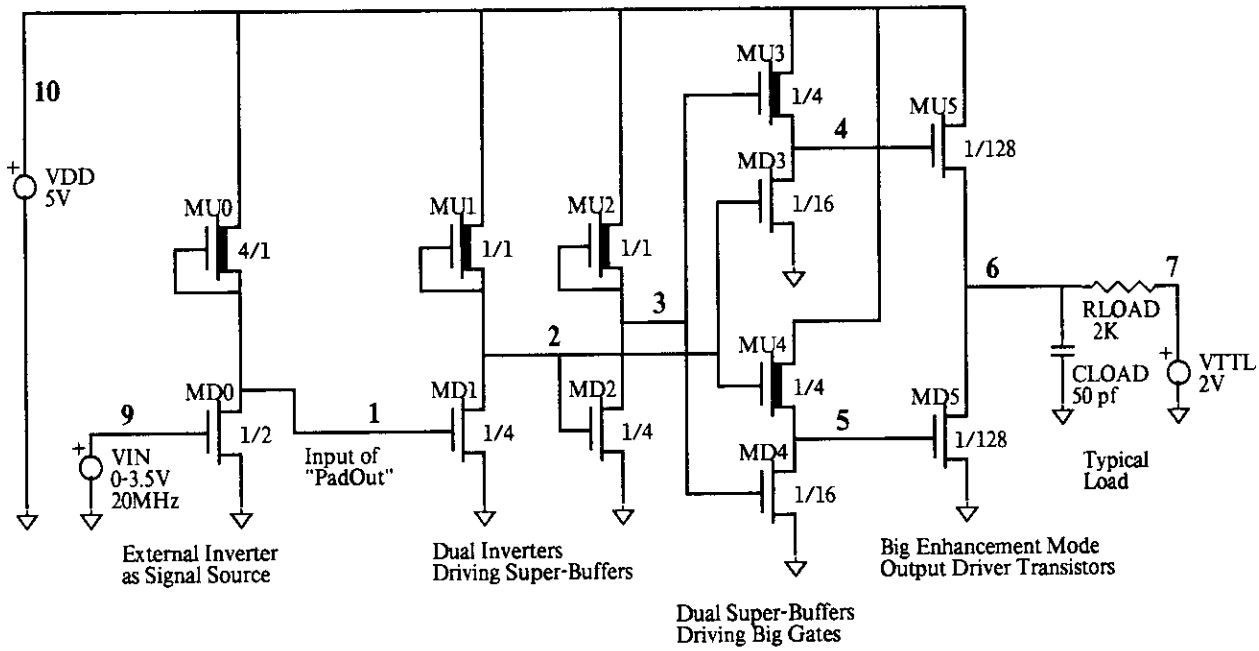


Figure 2.6.2.

Diagram of "PadOut" Output Pad Driver
with element names and node numbers
for simulation with SPICE.

```
PAD DRIVER SIMULATION
* RF LYON -- JULY 13, 1978
*
VDD 10 0 DC 5VOLTS
VTTL 7 0 DC 2VOLTS
VIN 9 0 PULSE 3.5VOLTS 0VOLTS 2NS 2NS 2NS 23NS 50NS
*
MD0  1 9 0 0 ENH W=12E-4 L=06E-4 AS=144E-8 AD=144E-8
MU0 10 1 1 0 DEP W=06E-4 L=24E-4 AS=144E-8 AD=144E-8
MD1  2 1 0 0 ENH W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MU1 10 2 2 0 DEP W=06E-4 L=06E-4 AS=144E-8 AD=144E-8
MD2  3 2 0 0 ENH W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MU2 10 3 3 0 DEP W=06E-4 L=06E-4 AS=144E-8 AD=144E-8
MD3  4 2 0 0 ENH W=96E-4 L=06E-4 AS=600E-8 AD=600E-8
MU3 10 3 4 0 DEP W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MD4  5 3 0 0 ENH W=96E-4 L=06E-4 AS=600E-8 AD=600E-8
MU4 10 2 5 0 DEP W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MD5  6 5 0 0 ENH W=768E-4 L=6E-4 AS=4000E-8 AD=4000E-8
MU5 10 4 6 0 ENH W=768E-4 L=6E-4 AS=4000E-8 AD=4000E-8
CLOAD  6 0   50P
RLOAD  6 7   2K
*
.MODEL ENH NMOS (NGATE=1E20 TPS=1 XJ=1E-4
+ CGD=4E-12 CGS=4E-12 CGB=2E-12 TOX=95E-7
+ NSS=-22E10 NSUB=8E14 )
.MODEL DEP NMOS (NGATE=1E20 TPS=1 XJ=1E-4
+ CGD=4E-12 CGS=4E-12 CGB=2E-12 TOX=95E-7
+ NSS=80E10 NSUB=8E14 )
*
.TRAN 1.0NS 80NS
.PLOT TRAN V(1) V(2) V(3) V(4) V(5) V(6) (0,8)
.WIDTH OUT=72
.END
```

Figure 2.6.3. SPICE Input Deck for *PadOut*

The simulator SPICE was used at Xerox PARC, on the MAXC2 computer, which has no floating-point hardware; therefore, the execution of the Fortran program was blindingly slow. Figure 2.6.3 shows the *input deck*, an ASCII text file. The SPICE program, like most widely available programs, was written for the card-reader/line-printer/batch-computing environment which is found at the typical university computing center. Therefore, be careful of input formats; only 72 columns of 80-column cards are used – long lines use continuation marks in column 1, as in Fortran. The documentation is sparse, but keep in mind that you should not do anything you could not do on a keypunch, such as lower case letters. See *User's Guide to SPICE* by E. Cohen and D. O. Pederson, from U. C. Berkeley Dept. of Electrical Engineering and Computer Science.

In the input listing, each line is called a *card.* The first line is the *title card,* and lines starting with * are *comment cards.* Each *element card* names a component (the first letter of the name determines the element type, such as M for MOSFET), tells what nodes it is connected to (in order, such as drain, gate, source, substrate), and gives a few parameters (such as width and length in centimeters). There are also *model cards* and *control cards,* which will not be described here, but can be seen in the listing.

We have described in the element cards the circuit of Figure 2.6.2 (some of the parameters are estimates, such as AS and AD, areas of source and drain). The first inverter is not part of PadOut, but represents a typical signal source, which is in turn driven by a 3.5 volt, 20 Mhz square wave generator with 2 nsec rise and fall times.

The output file produced by SPICE from the input shown was too long to include here. The most interesting part of it is shown in Figure 2.6.4, the graph of the time response of the various nodes, which is plotted line-printer style by typing the node numbers in appropriate columns. To make it readable, take a bunch of colored markers and draw in the curves for the nodes of interest. You will see that the response from node 1 to node 6 is noninverting, with t_{PLH} of 13 nsec and t_{PHL} of 9 nsec, measured at a 2 volt threshold (or more nearly symmetrical at 11 nsec if measured somewhere below 1 volt).

Is PadOut really this fast? Probably not on most processes; the model cards used here have estimates of the Spice model parameters which were felt to be realistic, but which gave results that are probably too optimistic for most typical 1978 processes. The inverter-pair delay from node 1 to node 3 is seen to be 6 nsec, where the inverter ratios are $k=4$ and the fanouts are $f=5$ (actually 6 for the first inverter). The delay estimate according to [Mead & Conway 1980] is then $(k+1)f\tau = 25\tau = 6$ nsec, so we may conclude that we have simulated a process with $\tau = 0.24$ nsec (transit time), which certainly is optimistic. The actual performance of PadOut will have to be determined by test, and will depend on where it is fabricated; some lines would be three times slower than this simulation.

Circuit simulation can be very useful to the integrated circuit/system designer if it is applied to those problems that require it, but should not be relied on to verify the correctness of a complicated system design. In digital system design with a consistent design philosophy, it is usually possible to

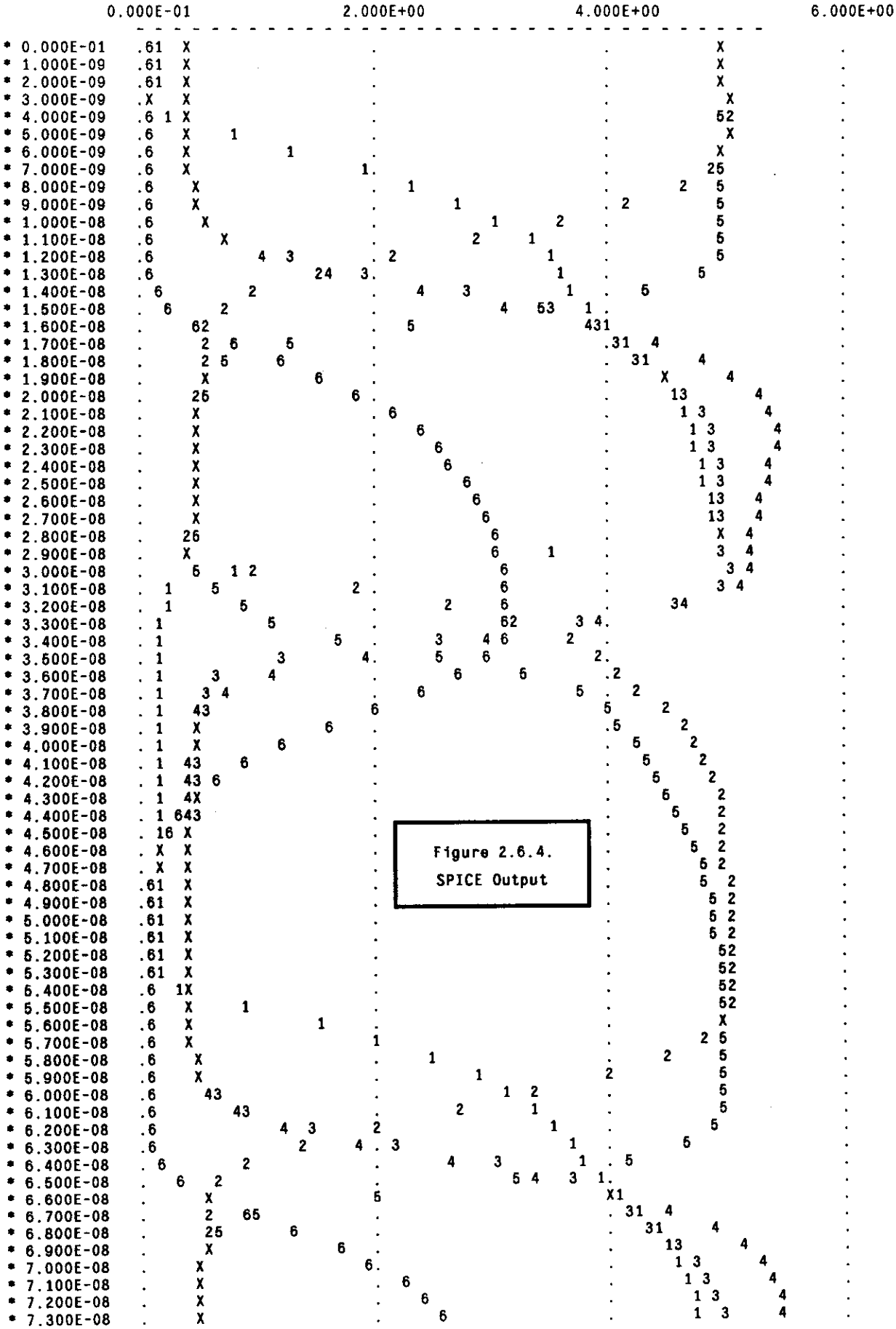


Figure 2.6.4.
SPICE Output

identify the critical parts of the design (for example the longest chain of pass transistors, the new RAM cell, or the node with the highest fanout); in this way, critical parts can be identified for simulation (see [Mead & Conway 1980] Chapters 1 and 7 for information on critical timing; see Chapter 4 for more on simulation and testing). Of course, even simulation will not verify that the design will run fast enough if the simulation parameters and models do not realistically reflect the process used to make the circuit.

IC designers have relied on simulation as a design tool for years. When the performance of a part being designed is critical (as is typical in manufacturing for sale), and the production/test turnaround is slow (also typical in the IC manufacturing business), circuit simulation is a necessity. However, in the topological design phase of a digital system, circuit simulation is not really helpful. By following strict design conventions and by employing relaxed design rules and conservative clocking schemes, it may be possible to design complete systems on a chip with only minimal use of circuit simulation. And, if turnaround is fast, an actual measurement may be a better way to determine performance than simulation is. A truly useful tool in this context would be a simple *logic-level* or *switch-level simulator*, working directly from the actual mask information [Bryant 1979]. Such a simulator, if able to handle the whole design at once, would be an invaluable help in finding logic errors and misrouted interconnections. The real task for the near future is to integrate simulation tools with design languages and layout programs.

2.7 Designing for Testability

Chip testing is an issue that should not be postponed until the wafers are delivered. While there exist sophisticated debuggers for pieces of software, there are no equivalent tools for chips. No hard and fast rules exist for designing a chip so that it can be effectively tested. One can only apply common sense and heed a few caveats.

A clean, modular design is a big asset in the testing phase. If the system is composed of a number of blocks, it may be possible to isolate each one as a separate project, complete with its own input and output pads. Thus, for instatnce, one could independently test the memory and the finite state control portions of a system. While the blocks themselves are not likely to be much use in their unbundled state, if each block is tested separately one need only correctly connect them together to construct a working system. The same principle applies to key cells in a repetitive design. If a novel memory cell is being tested, a single cell should be provided in isolation. The designer can then verify that the fundamental cells work (or not), even if the complete array doesn't. Testing small or moderately sized pieces of a system has the additional advantage that yield considerations are less significant.

Once the blocks are connected together it is still useful to have access to internal state information. One way to solve the problem is to provide internal test points (pads), with the

intention of probing them after the chip is packaged. This approach requires sophisticated probing equipment and is quite risky. The circuit and bonding wires surrounding the probe points are easily damaged if the probe shifts. Probe cards for such an arrangement are expensive, and it may not even be possible to construct such cards unless the layout of the probe points is carefully considered. A more appropriate solution is for the designer to provide standard output pads and drivers that monitor key nodes in the system. These pads are bonded in the usual fashion and allow easy access to the internal signals. Since there may be a number of signals of interest, the pad count may be reduced by using a single output pad driven by a shift register that is parallel-loaded.

Carefully designed input stimuli in conjunction with thoughtful circuit design can go a long way toward qualifying an IC. For example, in testing an ALU, a particular output response following the execution of a certain sequence of instructions may assure the designer that registers x, y, and z and data bus q are all functioning correctly. If each internal section of a system can be tested in this way, one can be confident in the correctness of the chip. In the event that the project does not work at all, process test patterns (see Section 6.2) containing simple transistors and inverters can provide reassurance that a minimum level of process quality has been achieved.

The importance of including support circuitry on chip should not be overlooked. With a system that requires multiple clock phases and several lines of input and output, massive quantities of time can be consumed debugging the tangle of wires and auxiliary instruments that comprise the test set-up. Such set-ups are fragile, subject to noise pickup and a serious liability when trying to measure the performance of systems. Simple support circuitry, for example a two-phase non-overlapping clock generator, can reduce the confusion to more tolerable levels. This assumes that the designer is not debugging the support circuits along with his design.

References

[Bell & Newell 1971]
C. G. Bell and M. Newell, *Computer Structures: Readings and Examples*, McGraw-Hill, 1971.

[Bryant 1979]
R. E. Bryant, "MOSSIM: A logic-level simulator for MOS LSI", MIT Lab for Computer Science, September 1979.

[Fairbairn & Rowson 1978]
D. Fairbairn and J. Rowson, "ICARUS: An Interactive Integrated Circuit Layout System", *Proceedings of the 15th Annual Design Automation Conference*, June 1978.

[Gibson & Nance 1976]
D. Gibson and S. Nance, "SLIC – Symbolic Layout of Integrated Circuits", *Proceedings of the 13th Annual Design Automation Conference*, June 1976.

[Johannsen 1979]
D. Johannsen, "Bristle Blocks: A Silicon Compiler", *Proceedings of the Caltech Conference on VLSI*, January 1979.

[Krause 1979]
J. Krause, "CIF Interface", ERL Report, Dept. EECS, U.C. Berkeley, 1979.

[Larsen 1978]
R. P. Larsen, "Symbolic Layout System Speeds Mask Design for IC's", *Electronics*, Vol 51, No. 15, July 20, 1978.

[Locanthi 1978]
B. Locanthi, "A Simula Package for IC Layout", Display File #1862, Computer Science Department, California Institute of Technology.

[Mead & Conway 1980]
C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA., 1980.

[Stone 1978]
M. Stone, "IC Design Under ICL, Version 1.0.", SSP File #1336, Computer Science Department, California Institute of Technology, February 1978.

[Williams 1977]
J. Williams, "Sticks — A New Approach to LSI Design", Master's Thesis, Massachusetts Institute of Technology, June, 1977.

CHAPTER 3
SILICON PATTERNING

MOS integrated circuits are constructed as a series of patterned layers on the surface of a silicon wafer. The generation of the *masks* used in patterning and the *wafer fabrication* process itself are complex, requiring special equipment and considerable expertise. The researcher who wishes to have his designs cast in silicon need not be concerned with all of the details of mask generation and wafer fab, yet he must be familiar enough with both to effectively deal with the vendors of those services.

3.1 An Introduction to Photolithography

The layers of a MOS IC are patterned by a *photolithographic* process. The layer-making process begins with the deposition or growth of some material, for example silicon dioxide, polysilicon, or metal, on the surface of the wafer. That material is coated with a thin layer of photosensitive chemicals, called *photoresist*, and exposed to ultraviolet light through a *mask*, which is a sheet of glass large enough to cover the silicon wafer. (The mask is coated on one side with opaque material that has been patterned to define certain areas on the particular layer.) If *negative photoresist* was used, those areas of resist that were exposed to light will be hardened, while *positive photoresist* is softened in the exposed areas. The exposure can take place with the mask pressed against the wafer (*contact photolithography*) or by projecting an image of the mask onto the wafer (*projection photolithography*). Projection techniques are becoming more widely used in spite of the extra equipment and maintenance needed since the masks are subject to less wear and contamination than contact masks. Consequently masks last longer and it is easier to control certain kinds of defects incurred in the photolithography steps.

Following exposure, the resist is *developed* by immersing it in a solvent that dissolves the unexposed (for negative resist) or exposed (for positive resist) portions, leaving the desired pattern. The patterned resist is hardened by baking at a low temperature and is then used to protect the covered areas of the wafer during the *etching* process. Two methods are common: in the older *wet etching* process the wafer is immersed in a bath of chemical etchant under controlled temperature conditions for a specific amount of time. Wet etching depends on the availability of an etchant that will dissolve the layer beneath the photoresist, yet not significantly attack the resist. For some materials, for example silicon nitride, the wet etchants dissolve photoresist as well as the desired material. Such materials require an intermediate pattern to be formed in another layer that serves as the actual etching mask. The intermediate material must be amenable to wet etching, yet resist the etchant for the layer beneath. In the case of silicon nitride, silicon dioxide is a suitable

intermediate layer. *Plasma etching*, a *dry etching* technique, utilizes a stream of ions and electrons to blast away material. Plasma etching gives better results for fine geometries and also permits the direct use of resist as an etching mask.

After the etching step the remaining resist is removed, leaving a pattern in the underlying material. This sequence is repeated for the various layers of the circuit. About six photolithography/etching cycles are required to build up a typical Si gate NMOS circuit. The entire process entails over forty individual steps, outlined in section 3.3.

3.2 Mask Generation

There are two readily available techniques for generating masks: *Optical* and *Electron-beam* (named for the the form of energy used to expose the plates). Both of these methods lead to a set of *master plates*, which may be used to pattern the wafers directly. More commonly, a secondary set of plates called *working plates*, is used in the actual photolithography process. Working plates are printed directly from the masters, thus allowing one set of masters to be used to produce many wafers.

Of optical and E-beam mask generation, optical mask generation is the older of the two processes, offering low cost and wide availability. The *pattern generation* process (see below) is slow, however, and its speed is directly related to the complexity of the design. Because of the difficulty in controlling alignment during the step and repeat process and the danger of reticle defects, it is costly to include more than two different chip types on the same set of working plates. Electron-beam mask generation is free of these shortcomings, but is not yet widely available. The flexibility and fast turnaround afforded by E-beam closely matches the requirements of many research institutions.

3.2.1 Optically Generated Masters

The first step in creating a mask is plotting the files provided by the designer on a photosensitized glass plate. This first plate, called a *reticle*, differs from a master or working plate in that it contains only one copy of the relevant chip layer and is plotted at 10x the actual size of the chip. The plotting process takes place in a *pattern generator*. Typical of such machines is the Mann 3000, which projects (*flashes*) the image of a variable size rectangle on the reticle. The input to the machine is the size of the rectangle, or *aperture*, the x and y coordinates of the center and the angle with respect to the x axis.

The nature of the reticle making process has a number of important implications for the designer. All shapes on the masks must be decomposed into simple rectangles. It should be noted that exposure time has a definite effect on the feature sizes on the reticle, in particular overexposed

areas tend to "grow" slightly; for this reason the designer should avoid substantial overlap between flashes. The pattern generation process involves complex mechanical motion; proper sorting of the individual rectangles of which the chip is composed can speed up the pattern generation process considerably – and thus lower the price (the bulk of the cost of optical mask generation is PG cost). For example a Mann 3000 PG machine is fastest at moving in the x direction, followed by aperture change, motion in the y direction, and finally, angle change. Unfortunately the optimum order is based on a complex function that depends on mechanical considerations as well as the pattern being flashed; in general this function is not known to the designer. Unless the designer has detailed knowledge about the PG machine being used, he is probably better off using a simple sorting algorithm (for instance lexicographic ordering based on what the particular PG machine is fastest at) than trying to second guess the pattern generator.

Optically generated masters require several special features on the reticle that are used during intermediate steps in mask making. A *parity mark*, consisting of an arrow or triangle, is sometimes included on each mask layer to help the operator orient the mask. The mark is placed outside of the boundary of the chip pattern. *Fiducials* are small crosses which also appear on each layer outside of the boundaries of the chip. These are used in the *step and repeat* process (see below). Often the parity marks and fiducials are provided by the mask house thus making it unnecessary and undesirable for the designer to supply them. Parity marks and fiducials appear only on the reticles and not on the finished master plates.

The reticles are used to make a set of master plates in a step and repeat machine that projects an image of the reticle (reduced 10x) onto a photosensitized plate. By precisely stepping the image across the master a matrix of images of the reticle is created. The fiducials are used to control the distance between exposures and to align the reticle images relative to one another. It is possible to interstep two different reticles on the same master, but it becomes increasingly difficult as the number of reticles goes up. We have not found any manufacturers willing to guarantee alignment specs for more that 3 reticles.

3.2.2 E-Beam Masters

As in the optical process, electron-beam mask generation equipment can be used to create reticles that are stepped and repeated on a master plate. More commonly, however, an entire master plate is written in one step. E-beam masks offer several advantages to researchers interested in fast turnaround: one-step mask generation (if the masters are used to directly expose the wafers), speed, flexibility, and reduced defects from certain causes. For instance, a defect on a reticle means that each and every chip will have the same defect, in addition, the step and repeat process is a potential source of defects (for example, alignment problems, defects from dust specks). Both of these problem areas are eliminated with e-beam masters.

Unlike optical pattern generation equipment, electron-beam exposure systems are raster oriented. The mask can be visualized as a piece of graph paper, where the squares are the same size as the e-beam diameter (typically 0.25μ or 0.5μ). All geometric data is ultimately converted into a *bitmap* (a rectangular array of 1's and 0's), which is placed on top of the graph paper — the squares containing 1's are exposed, those containing 0's, not. Conceptually, the exposures are made by sweeping the electron beam in a repeating "S" pattern from the lower left-hand corner of the mask, *blanking* and *unblanking* the beam according to the input stream of bits. To a first approximation, the beam visits each point on the mask regardless of whether the point is exposed, and so the *writing* time is independent of the design complexity. (In practice, this is not entirely true. Some machines are programmed to skip large blank areas, and so take less time to write sparse designs.)

For practical reasons, the writing sequence is not quite that straightforward. Assume that we wish to write and array of 8 identical chips (refer to Figure 3.2.1). The chip is divided into horizontal strips of fixed height and the geometric shapes within each strip are fractured into rectangles and trapezoids (or approximated by same). Software is available to convert conventional PG formats to this e-beam format, or the designer can generate the trapezoids and rectangles directly. The location of each strip of the chip, in this case there are four strips, along with other information is used to create a command sequence for writing the array.

The first step in writing is converting the trapezoids and rectangles for a given strip into a bitmap, this process, called *corefill* (because the bitmap is loaded into core) is relatively time consuming. For this reason, it is only done once for a given strip. The machine then writes every area on the mask that is covered by that strip, before it converts another. In our simple example the machine would write identical strips a,b,c,d,e,f,g,h in that order, then convert the next strip and continue the process. Mechanically, the mask (affixed to a *stage*) is moved in the x direction, while the electron beam scans in the y direction along short *scan lines.*

For more complex arrays, the only penalty paid is in corefill time, since the writing time is more or less constant. If chips 1,3,5,7 (the odd group) are identical to each other but different from the chips in the even group the machine might first corefill with the bottom strip of the odd chip. Strips a,d,f,g are written in that order. Corefill would proceed with the bottom strip of the even chip, and then strips b,c,e,h would be written. Using this technique, have combined as many as 8 different chip types on the same set of masks. Such an undertaking would be impossible if optical masks were employed. Aside from the great expense of generating 8 reticles, each reticle would have to be perfectly aligned through 8 step and repeat cycles.

3.2.3 Working Plates

When needed, working plates can be made from the masters by contact printing. In cases where a large number of working plates are required the mask house may make several sets of

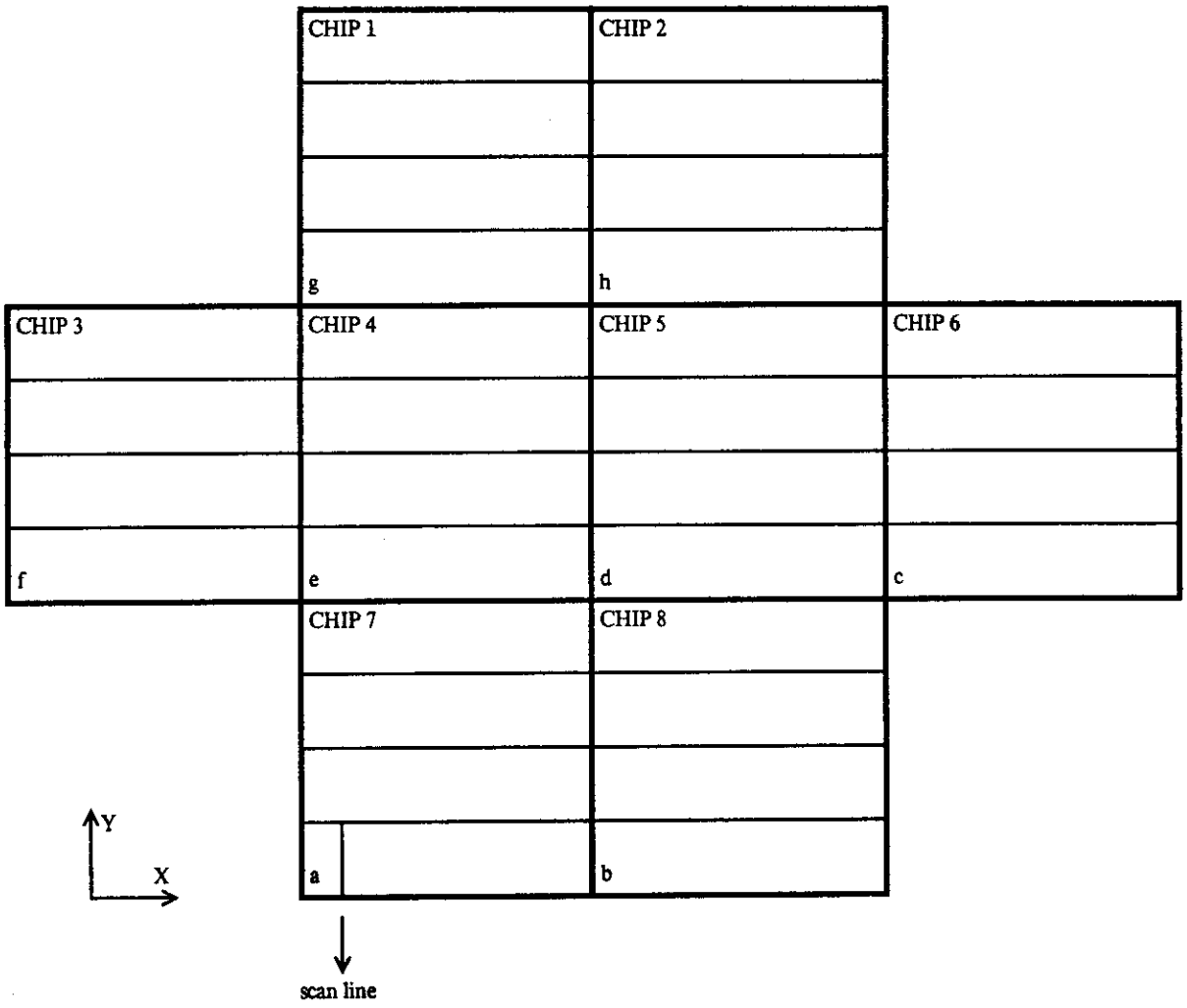


Figure 3.2.1 E-Beam Writing Sequence

(after [Varian 1979])

submasters and print the working plates from them.

To improve the quality of the masks, reduce cost, and shorten turnaround time, it may be possible to use the masters directly for wafer fabrication. This approach is particularly attractive with masters generated on electron-beam exposure systems since it reduces mask making to a one-step process with a possible turnaround time of a few days. Once the decision is made to use master plates directly, it is generally not possible to have working plates made from those masters. This is because the copying process causes the dimensions on the copies to differ from the masters, so the mask house must compensate for these changes in advance (i.e. as the masters are being made).

3.2.4 Mask Specification

The researcher is faced with the task of specifying many details so that the masks will be made correctly. Appendix A contains copies of instructions that we have sent to mask houses, providing an overview of the type of data needed. It is essential that those wishing to have wafers fabricated understand the mask generation and wafer fabrication processes in enough detail to make reasonable decisions. The following paragraphs briefly cover some tradeoffs and decisions the researcher must make.

Mask *polarity* — whether the plates for each layer should be *opaque field* (clear features) or *clear field* (opaque features) – must be specified by whomever orders the plates. The requisite polarity for each mask is specified by the fab line; typically a mixture will be called for. The choice depends on the process step and the type of photoresist used. The photoresist is selected partially on the basis of requisite linewidths for the fabrication process. More important, however, is the field area involved with the particular plate. A speck of dust on an otherwise clear area of the working plate will cause a pattern to be made in the photoresist. If negative resist is being used the speck will make a hole in the resist that will enlarge somewhat due to undercutting in the subsequent etching step. Positive resist will leave a small dot where the speck was; this dot will probably be etched into oblivion. The fabrication line decides which of these factors to trade off in choosing the polarity of the working plates.

The opaque material covering the plates may be photographic emulsion, iron oxide, or chromium. Emulsion is the least expensive but relatively easily damaged in the contact photolithography process. Chromium and iron oxide give better line resolution and are very hard, but they are more expensive. The glass plate itself is available in various grades, the least expensive being *"green glass"*. *Low-expansion glass* yields masks that are more stable, but may run several times the cost of green glass. There are also several options in glass thickness. All of the alternatives are provided so that users may choose masks that will provide the best performance (which includes cost, defect density, repeatability, etc.) for their application.

The plate specifications may be dictated by the fabrication line, as they may be used to working with a particular type of plate. In this case the researcher has little choice but to order (and pay for) whatever the fab line requires. In the event that the fab line has no preference, a low-cost option may be quite adequate, since masks for a research chip set are rarely used for more that one run of wafers.

Often the mask house or fab people require some special features to be included on the mask set. *Critical dimensions* (CD's) are simple lines or crosses of a fixed size appearing on each layer; they are used by the mask house to adjust exposure and developing time to insure that these marks and hence other features on the mask are the correct size. Additional features to be put on each mask may include an identification code for the process step. The fabrication line may have specific codes that they wish placed on the masks. In order to register the layers during wafer fabrication, a set of *alignment marks* (see section 4.3) must be included on the masks. The alignment marks may be a specific set for the fab line or may be designed by the researcher.

The seemingly simple concept of mask *parity*, which specifies up from down and left from right, has turned out to be quite confusing to keep straight in practice. A number of factors confound the issue:

1. Some mask generation equipment uses a left handed coordinate system (e.g. Mann) others, right handed.
2. The initial data may be reversed (mirrored) a number of times, depending upon the details of the process used to arrive at the working plates. Thus, working plates may come out reversed, even though the input data was not.
3. The same mask manufactures may make masks for digital and analog circuits in MOS, bipolar and other technologies. Thus the natural frame of reference for the designer is largely irrelevant to the mask house, and he should not count on the mask people having any intuition about the orientation of shapes on the masks.

We have found that the most effective way to specify mask parity is to include some text in the same location on each layer, and to instruct the mask house as to how the text should appear on the final plates (be they working plates or masters). Usually up and down is not an issue (the chip can be rotated 180^{0}), so two things must be specified. The first is whether the text is to appear *right reading* or *wrong reading* (normal or reversed), and the second is which side of the plate the text should be viewed from (chrome side or non-chrome side). A MOS wafer processed with masks whose text is WRONG reading when viewed from the chrome side will have features normally oriented. Other schemes for specifying parity are possible, for example arrows that "point upward and left", but few are as unambiguous as a string of text.

It is important to verify the correctness of masks as much as possible. When optical mask generation is used, it is usually possible to order color enlargements of each reticle; these *blowbacks* are typically about 100x-150x actual (chip) size. The layers can be checked individually and in combination by superimposing the films on one another. *Black and clear* transparencies (usually

8½" x 11") may also be made at the same time. They are sometimes used in the interaction between the operators on the fab line and the designer to indicate the location of features on the mask such as alignment marks. At the present time film blowbacks are not generally available from E-beam masks. Large checkplots are the only recourse and provide a reasonable means of checking individual layers, but are not particularly helpful in checking combinations.

Before the mag tape can be sent off to the mask house some information must be obtained from the fabrication line regarding their process. Varying etch conditions may cause the fab line to request that features on the masks for certain layers be altered (i.e. stretched or shrunk) by a constant amount, for example 0.5 micron around any border, in order to produce the desired dimensions on the silicon. These dimensional adjustments can be made in one of three ways:

1. The circuit designer can be required to change his design to take into account the over- or under-etching at the fabrication line. This entails considerable work on the part of the designer each time the circuit is implemented on a different fab line, but has the advantage that the designer retains complete control of the layout geometry.
2. Software could be provided to input the original design file and produce a new design file that had the borders of features expanded or contracted in the appropriate way. This approach may require the use of complex algorithms in order to correctly modify the original file since minimum spacing design rules may be violated by enlarging adjacent features while gaps and discontinuities may be introduced by shrinking features which abut in the original design.
3. The mask house may be able to effect the changes by adjusting exposure time and other parameters in the mask generation process.

Once all of this information has been collected and reduced to a set of files on mag tape and some written instructions, the mask house takes over. When the masks are returned they are passed in turn to the fabrication line along with more instructions. The total elapsed time for mask making and wafer fabrication can be 8-12 weeks. During this time the designer should be preparing for the day when finished wafers are delivered.

3.3 Wafer Fabrication

A number of wafer fabrication lines offer "standard" n-channel Si-gate MOS processes. In principle, one need only produce a set of masks that is compatible with the fab process and about $3,000, and the fab line will deliver a *minimum run* (about 20) of finished wafers. The availablility of these processes allows IC designers to take a black box view of NMOS processing.

Complementary MOS may soon reach the same "stable" state. Today several firms offer CMOS processes, but they vary widely. In addition, many low-level tradeoffs may be made in

CMOS, for example a particular gate may be implemented in complementary form or using n-channel or p-channel devices exclusively. Another problem arising in current CMOS processes is that the layout that most cleanly reflects the functional topology of the system often has to be distorted in order to move transistors of the same polarity close together. This is so that they can be placed into an area of the same substrate polarity. These factors make it difficult to design independently of the fab line, and to provide an overview of the process.

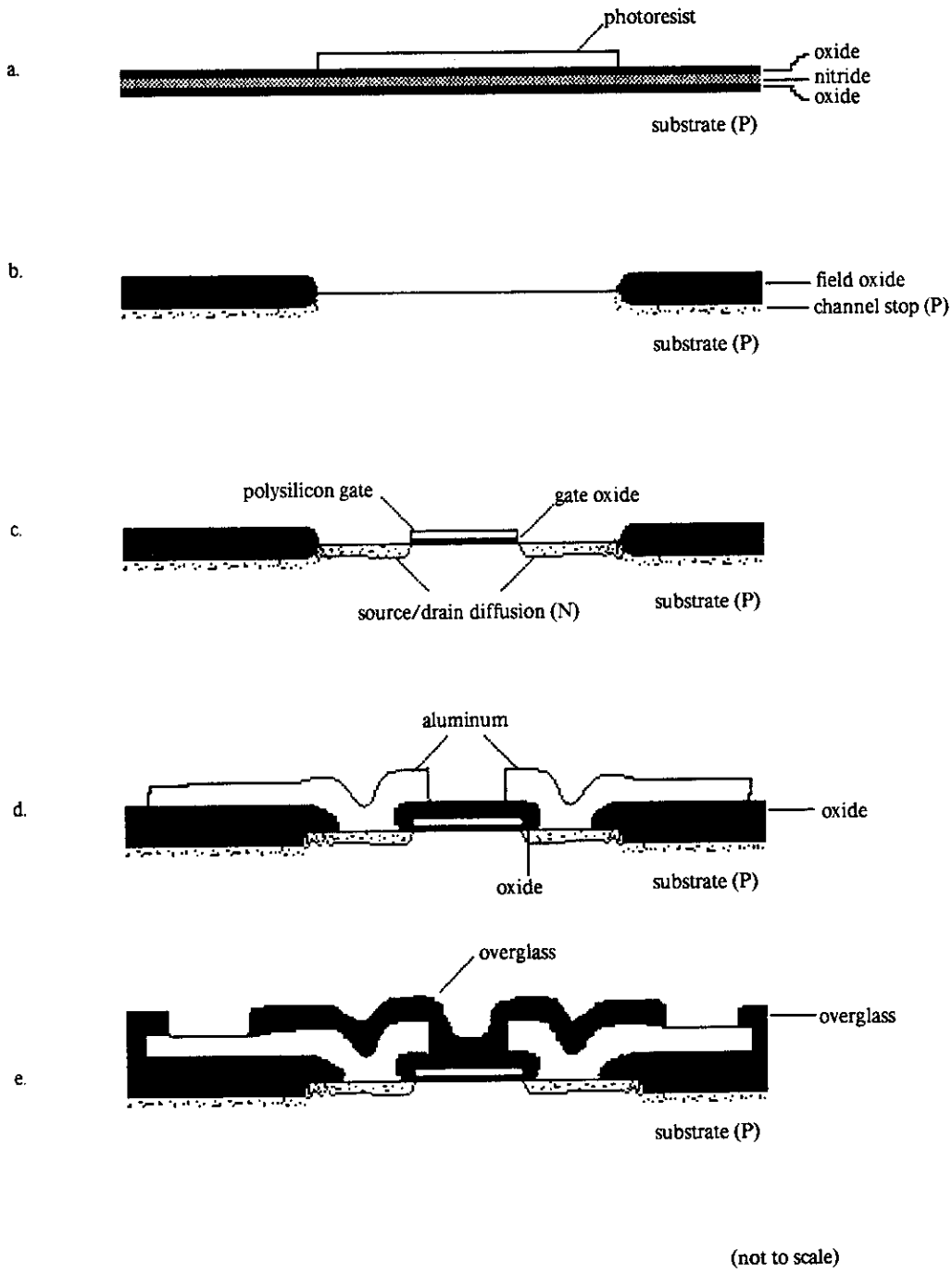
3.3.1 The Si-gate NMOS Process

The process discussed here is a standard Si-gate n-channel MOS process, such as available from a number of fabrication lines. The reader need not be concerned with learning all of the details of the process, indeed most of the decisions concerning processing are made by the fabrication line. The designer may not know which particular techniques the fab line uses, and may not care as long as his standard circuits exhibit normal performance. This section is presented to provide background for those interested in what really happens behind the clean room doors. It may be helpful for the reader to make a sketch of the state of the wafer after each step as this section is covered.

The wafer of *p-type* (*100 crystal orientation* for lowest interface state density) silicon is scrubbed and a thin layer of silicon dioxide (hereafter called "oxide") is thermally grown on the surface. This layer serves as a mechanical buffer zone for the silicon nitride (Si_3N_4) that follows. The buffer zone is needed to relieve stress caused by differences in the coefficients of thermal expansion of silicon and silicon nitride. A layer of Si_3N_4 is deposited by *chemical vapor deposition*, then another layer of oxide is grown. Photoresist is applied over the entire surface and the wafer is exposed to ultraviolet light through the *diffusion layer* mask. The resist is developed, leaving open areas over the *field* region (figure 3.3.1a). The top layer of oxide is etched away wherever there is no photoresist using a hydrofluoric acid solution. After the resist is removed this top layer of oxide is used as a mask for patterning the nitride since the photoresist alone will not stand up to the chemicals used in the wet etching of silicon nitride. A third etching step is used to remove the bottom layer of oxide. *Ion implantation* is used to place the *channel stop* region and a thick *field oxide* is grown over those areas. The field oxide and the channel stop are *self-aligned* with respect to the source/drain diffused areas (the nitride covers the source/drain areas during channel stop implant and prevents oxidation of the underlying silicon during field oxide growth). The remaining nitride and the thin oxide under it are removed resulting in the profile shown in figure 3.3.1b.

Next a layer of photoresist is applied and the wafer is exposed through the *depletion mode implant* mask. The resist is developed, leaving open spaces in the gate regions of the depletion load transistors. Another ion implantation step occurs here (using the resist as a mask) to alter the threshold voltages of the depletion load transistors. The resist is removed and a thin layer of *gate oxide* is grown. If there are *buried contacts* used in the IC design more photoresist is applied, the wafer is exposed through the buried contact mask, the resist is developed, the gate oxide is etched

Figure 3.3.1 Si Gate NMOS Processing Steps



(not to scale)

away in the contact areas, and the resist is removed. This allows the *polysilicon gate* material to contact the substrate in selected areas.

A layer of polysilicon is deposited from a chemical vapor and a thin layer of oxide is grown on top of that to provide a surface that photoresist will adhere to. Resist is applied and the wafer is exposed through the polysilicon layer mask. The development of the resist leaves the gates of the transistors covered; the uncovered areas of oxide and polysilicon are etched away (a little field oxide is also removed). After the resist is removed the *source* and *drain* regions are doped (figure 3.3.1c) in a phosphine gas atmosphere. Since the edges of the polysilicon gates define where the source/drain regions begin these features are also self-aligned. Here self-alignment results in a significant reduction in parasitic capacitance due to the near zero gate to source/drain overlap. A thick layer of oxide containing P_2O_5 is deposited over the surface of the wafer. This layer is reflowed for better coverage of the steps in the surface and a layer of photoresist is applied. *Contact hole* areas are defined using the contact cut mask and the oxide is etched away where the metal layer will contact the underlying features. After the resist is removed the source and drain are doped again (this is to prevent a phenomenon called *spike-through* – essentially shorting of the aluminum contacts and the substrate through the shallow source and drain regions). A layer of aluminum is evaporated onto the surface of the wafer, followed by the application of more photoresist. Exposure (and subsequent development) through the metal layer mask leaves resist protecting the metal runs and contacts. The uncovered aluminum is etched away and the resist is removed (figure 3.3.1d). The wafer is then *annealed* (heated at a low temperature) to remove radiation damage resulting from the electron beam that is used to heat the aluminum during the evaporation process.

A thick layer of oxide is deposited on the entire surface of the wafer to provide physical protection. Windows to the bonding pads are etched through this layer in another photolithography step using the *overglass layer* mask. At this point (figure 3.3.1e) the wafer is finished, ready to be broken apart, bonded and tested.

References

[Varian 1979]
Varian/Extrion Division, "Ee˙BES-40 Electron Beam Lithography System", Varian Corp., Gloucester, MA., August 1979.

CHAPTER 4
PRACTICAL CONSIDERATIONS IN IC PATTERN PREPARATION

Implementing custom integrated circuits is an expensive process, both in terms of dollars and effort. The monetary expense may be so high that it is impractical for a designer working on a research budget to have chips made for a single circuit or system. One answer is to amortize the cost of the masks over several projects by putting more than one design on a single chip. If as many as ten different designs can be placed on a single 1/4 inch square chip, the cost of masks may only increase from about $7000 to $8500. A minimum run of about 20 3" wafers (Si gate NMOS) costs roughly $3000 at any of a number of fab firms; such a run yields 1600 chips. The cost in manpower is similarly high – it might require one man-month to administrate such a multi-project chip, in addition to the design effort. Good design tools can reduce the human effort necessary, but of course building the requisite tools is a time consuming process. Luckily, in many research groups manpower is easier to come by than real dollars.

This chapter focusses on the process of creating multi-project chips (*MPC's*).

4.1 Merging Many Projects

Over the past two years much progress has been made in streamlining the process of merging many individual IC designs into the specification for a single mask set. Early efforts were carried out in and *ad hoc* fashion, with many costly errors made. With each multi-project chip, techniques and support systems have been improved. A recent chip set, called MPC79, was organized by researchers at PARC-SSL using a new form of "VLSI implementation system" that enabled the remote-entry of designs by a large community of designers scattered throughout the country. MPC79 contained 82 designs on 12 die types, and its total implementation time was 29 days. For more details on that chip set see [Conway, et al 1980].

At least in the near future, most first-time multi-project chips will be assembled by hand by a few coordinators. As in any endeavor involving many participants (the designers), there are a great number of purely administrative details to be attended to. It is imperative that a convenient means of *communication* be available. Messages and design files must be transported on a regular basis, and as final deadlines approach, the lack of a rapid means of communication can become an insurmountable barrier. In a small, intra-departmental effort, written messages and a central design facility are probably adequate. For an MPC involving geographically diverse locations and differing design systems, some form of electronic message and file transport capabilities are essential.

In the remainder of this section, we will outline the steps of the merging process, elaborating as necessary.

1. Creation and checking of design files

   This process may be left up to the individual designers, or may be monitored by the coordinators. Large efforts, like MPC79, favor the former approach, while smaller chip sets allow the coordinators to actively aid in the checking process. In the absence of design rule checking programs, designers should be assigned to cross-check someone else's design. Many trivial errors can be eliminated in this way. We have found that there must be a *central depository* for design files, where the official version of each design resides. Participants are only allowed to change files with the consent of the coordinators, who are responsible for any actions that an update requires. These activities include such things as updating logs, running new checkplots, or perhaps notifying human design rule checkers that a new version is in place.

2. Submission of final designs

   If the coordinators have been involved in checking designs, this step simply means freezing the state of the central depository. If not, when the appointed deadline is reached the designs should be collected and moved to a protected location so that no more changes can be made.

3. Conversion to standard format

   Once the designs are frozen, they must be converted to a standard format (e.g. CIF 2.0). Alternatively, the coordinators may require that designs be submitted in a standard format, perhaps with other restrictions applied (for example, they may require that each design have the lower left-hand corner located at 0,0). From each standard design, information about location and bounding rectangle can be obtained.

4. Merging

   The frozen designs are now packed into one or more chips (see Section 4.2 below). Typically 3 to 10 projects will fit onto one chip; each must include a number of standard features (see Section 4.3), which are merged in as any other project. Designs expressed in a graphics language are easily merged by translating and rotating the various designs, and then concatenating the design files (note that symbols may have to be given unique prefixes in order to avoid naming conflicts).

5. Conversion to mask format

   Each multi-project chip is now available as a single large design file. These files must be converted into PG or e-beam format and written on a mag tape. At the same time, features may be over- or under-sized to meet fab requirements (see Section 3.2.4). Instructions detailing die sizes, location, step and repeat distances, mask materials, and more (see Appendix A) are written. They, the mag tape, and a check are sufficient to have wafers made.

4.2 Physical Constraints

The designs are arranged to minimize the area of the chip bearing in mind a number of important factors. Optical equipment limitations at the mask house make it difficult to generate masks for chips larger than 10mm by 10mm. Defect-free reticles become harder to generate as the chip size increases, thus it is disproportionately expensive to make masks as the chip gets large. If e-beam masks are available, these size restrictions are not so critical. The 10mm x 10mm size limit is somewhat misleading because of an additional restriction imposed by current packaging technology. The *cavity size* of a standard 40 pin dual inline package (DIP) is about 7.5mm x 7.5mm, thus projects should be limited to this size unless there is access to special packages. The 10mm x 10mm chip must be subdivided to meet this constraint by placing *interior scribe lines* between projects; these scribe lines must extend all the way across the chip (and thus across the wafer), that is, interior "tees" are not allowed.

When packing the designs, it is wise to consider the wire-bonding apparatus that will be used to connect the circuit pads to the package pins. Depending on the type of equipment used (see Section 5.3), it may be difficult to wire bond configurations where wires must run long distances or at acute angles. A particularly troublesome layout places a small project in the middle of a large chip. In this case long wires must be run to the periphery; they are subject to sagging and subsequent short circuiting. Long wires also protrude above the top of the package, because of the trajectories that they must follow.

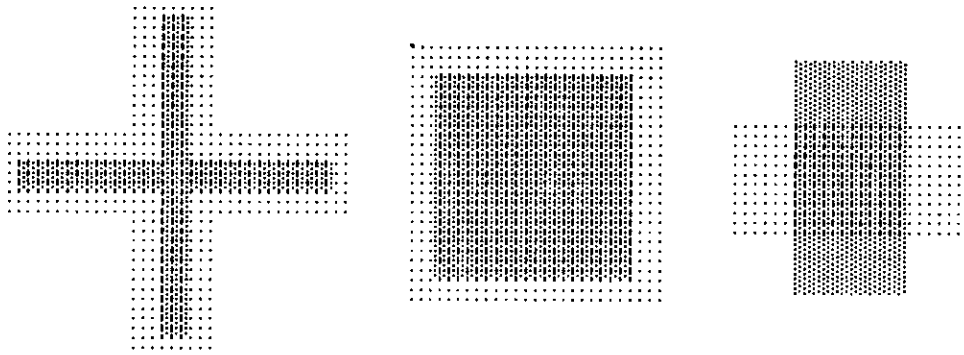
The overall size of the chips is impacted by the expected *yield*, which is the fraction of the IC's that function correctly. As the *active area* (the area containing active devices but excluding empty space, bonding pads, etc.) grows the yield decreases geometrically. Typical yields for a 6mm by 6mm circuit, assuming standard defect densities, are about 20-40%; in industry, yields much below this figure are not acceptable profit-wise. Designers in a research environment may well be able to tolerate low yields since a project is rarely dependent on all of the devices on the chip working. Huge projects may still be feasible, as a yield of a only few percent gives the designer enough chips to verify his design, measure the performance and demonstrate feasibility.

4.3 The Starting Frame

At some point the designer has several complete IC designs that are ready to be turned into chips. Before his design can be realized, several important details have to be taken care of which are not part of the actual circuit design process. Among these are the physical placement of several projects and test patterns on the multi-project chip plus the the addition of some extra features required by the fabrication line. Test patterns are useful for an evaluation of the quality of the wafer processing, for checking standard circuit parameters, as well as post mortem debugging should a chip fail to perform correctly. Most of these relatively fixed, universally required features (e.g.

CD's, fiducial and parity marks, alignment marks and scribe lines) can be collected at each research site and grouped together in a *starting frame* (see also [Mead & Conway 1980]). This starting frame provides a set of "symbols" (or whatever construct is appropriate in the local design system) that can be combined with the individual design projects to provide the masks for a complete multi-project chip.

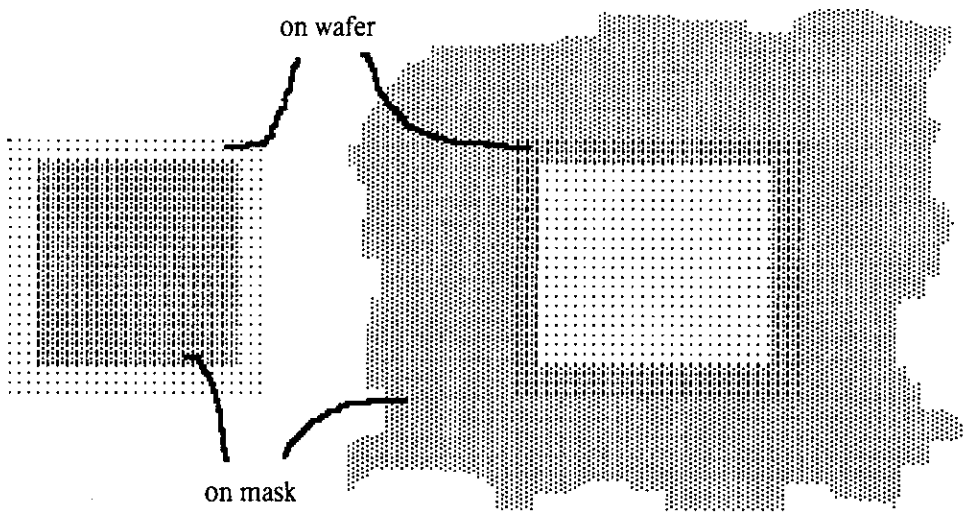
The most important features that must be added to the net circuit is the set of alignment marks, which are needed to register subsequent layers on the IC with one another. Alignment marks take many forms:



light area on wafer dark area on mask

– but their purpose is the same. There is little magic in designing alignment marks; in fact, almost any reasonable features will do. However, a carefully designed set can mean the difference between good devices and those that are only marginal.

When the designer is deciding on which alignment marks to use it may help to consider the following scenario. The fabrication line operator puts a partially processed wafer onto the movable (x, y, rotation) stage of the alignment machine. The next mask is held over the wafer and the operator looks through a microscope from above. First of all, is it possible to locate the alignment marks? The designer can help by providing "black and clears" on which he has indicated the location of the alignment marks. A line or box enclosing the marks may also draw the operator's attention amidst the confusion of the other features. After the marks are located, is it possible for the operator to successfully align with them? Consider the case where the alignment marks consist of a large square on the wafer and a small square on the mask.



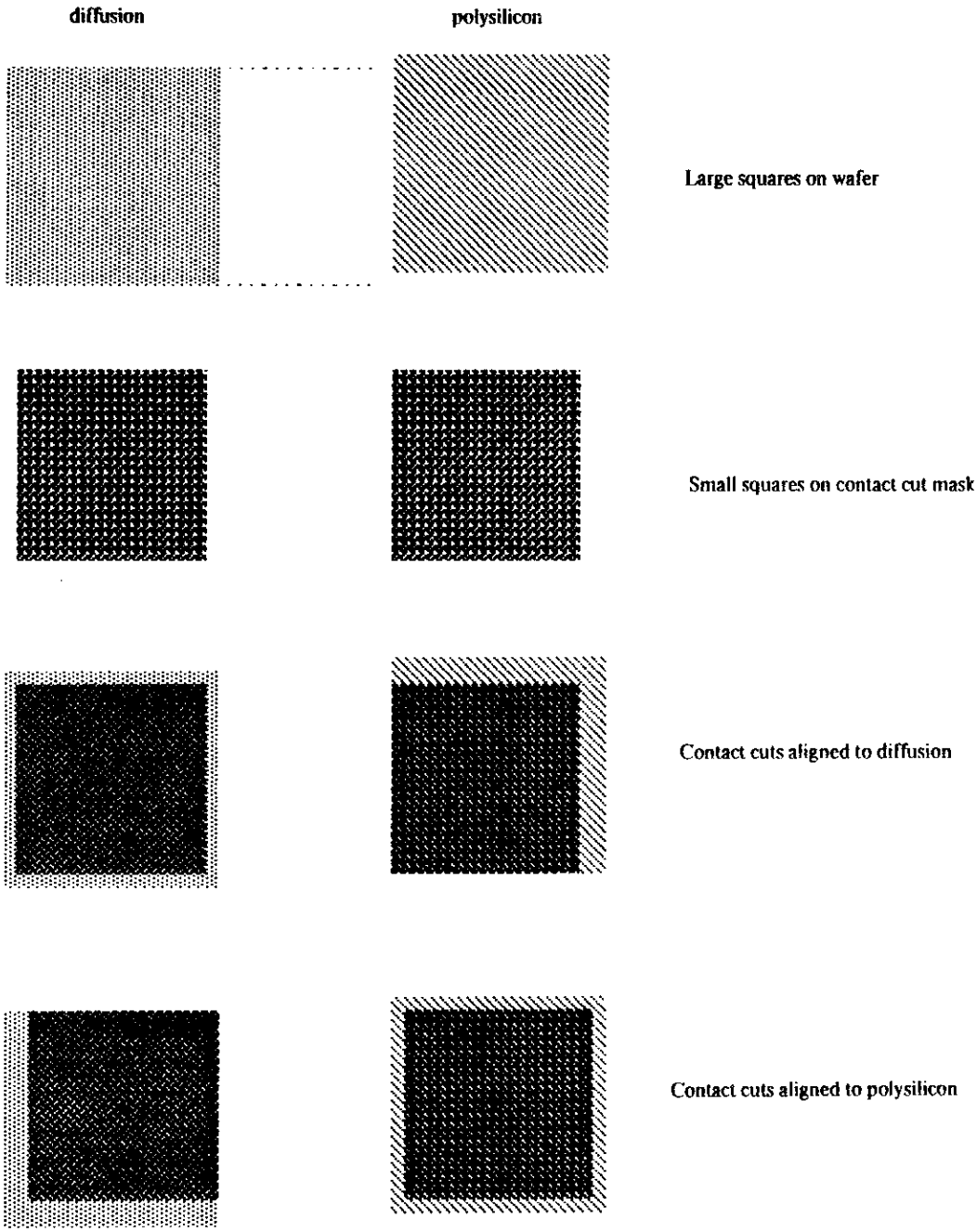
Clear Field Dark Field

The operator is supposed to center the small square over the large one. This system is fine if the small box is opaque on a clear field (see the explanation of working plate polarity in Chapter 5), but not when the small box is clear on an otherwise opaque mask. In the latter case the designer should have an alternate version of the alignment marks for opaque field masks, then one or the other set will work for the mask polarity used in the particular fabrication step. Another alternative is to design a set of marks that can be used regardless of mask polarity.

As the operator tries to line up the alignment marks, is it obvious which small square goes over which large one? A one square shift is certain to be disastrous. Some type of *gross* alignment mark should be provided to prevent shifting; again there are many alternatives – an enclosing box or a simple square which is superimposed over one already on the wafer, or even numbering the small/large square combinations. Furthermore, it is important to eliminate ambiguity regarding the layer to which the current mask is aligned. For example, in figure 4.3.1 there are two large squares in place on the wafer, one in diffusion, the other in polysilicon. The operator has two small squares on the contact cut mask to line up over the two large squares. Unfortunately the large ones are not exactly in line because of a small misregistration introduced in a previous step. The operator must decide whether to align to the diffusion or the poly feature, or perhaps to split the difference between the two. Whichever course the operator chooses may affect device operation. The *designer* should make this decision by providing only one pair of marks. (The actual set of alignment marks chosen for our starting frame are discussed in section 6.1.)

After the integrated circuits are fabricated one must determine whether or not they are functioning correctly. While the designer has the option of simply powering up his circuit and seeing if its input/ouput behavior is correct, a more satisfactory test method might use several *test structures* included on the chip (see Chapter 5). Simple structures like inverters can answer yes/no questions (Were the wafers processed to a minimum level of competence? Do individual transistors work?) and thus indicate whether more complete testing is warranted. More importantly, test

Figure 4.3.1 Alignment of Contact Cut Mask



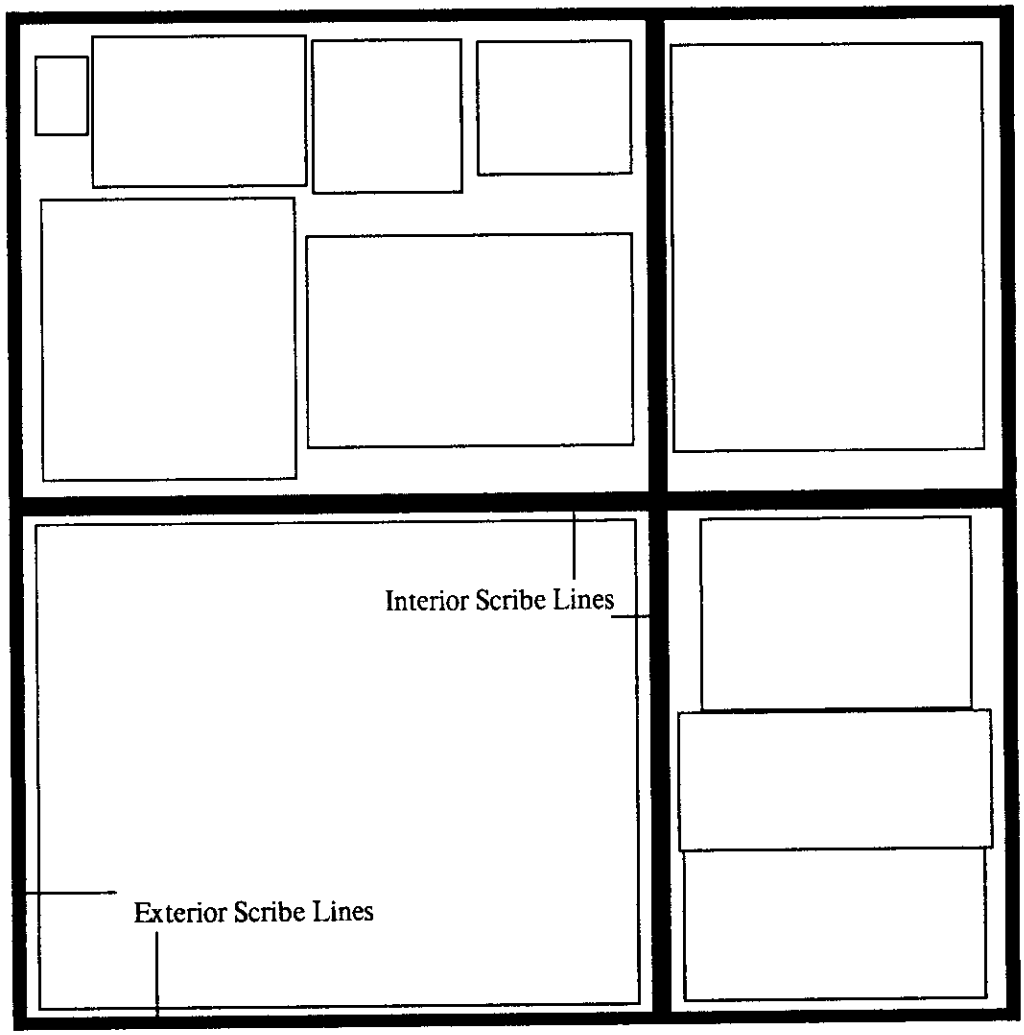


Figure 4.3.2 Overall View of a Multi-Project Chip

patterns can provide information which is useful in determining why a batch of chips does not work, or performs poorly. Properly designed test patterns can show wafer processing problems, or eliminate this cause, narrowing the search to the area of design errors.

Ultimately the various designs and the starting frame have to be combined into a single IC description. This involves merging of a number of files, usually in a geometric design language, into a single file containing all of the integrated circuit designs. Fiducials and parity marks may need to be added outside of the area occupied by the designs and the starting frame. The chip pattern is repeated on the surface of the silicon wafer many times; 2", 3", and 4" wafers are commonly available – a 3" wafer holds about 45 10mm by 10mm chips. *Exterior scribe lines* (see figure 4.3.2) are placed around the periphery of the area occupied by the project set. The purpose of the scribe lines is to provide a "lane" down to the silicon substrate in which the diamond-tipped scribe tool (see Section 5.2) will ride. The wafer will be broken into chips (also called *dies*), as defined by the scribe lines, following fabrication.

References

[Conway, et al 1980]
L. Conway, A. Bell, M. Newell, R. Lyon, R. Pasco, *Implementation Documentation for the MPC79 Multi-University Multiproject Chip-Set*, Xerox-PARC Report, January 1980.

[Mead & Conway 1980]
C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA., 1980.

CHAPTER 5
WHEN THE WAFERS ARE DELIVERED

The return of the finished wafers is an exciting moment for the IC designer. Proper preparation for this day can help to prevent frustrating delays in *wafer separation, packaging* and *testing*, thus promoting rapid feedback concerning the success of the designs. Testing complexity increases even faster than design complexity. Thus testing needs suitable preparation as already mentioned in Section 2.7. If "multiple iteration with fast turn-around" is to become a successful IC design methodology, testing must also be efficient and streamlined. Often the designer tends to relax after the masks have been submitted, when instead he should immediately begin preparation of the test set-up for his chip. The following discussion of wafer separation, chip bonding and testing techniques is primarily aimed at those in a research, rather than a production environment.

Testing progresses in a multistep, hierarchical approach. First it has to be determined whether the wafer was properly processed. This should be done on the uncut wafer, since there is no point in wasting the effort of chip separation and mounting on defective wafers. For large designs it may be worthwhile to do some preliminary electrical tests by probing individual chips in order to determine which ones should be mounted.

5.1 Process Testing

Some tests can be performed visually by inspecting the wafers under a microscope. All wafers should be checked for a clean look, for lack of obvious defects such as scratches or missing parts, for sharply defined patterns in all visible mask levels, and for the absence of obvious shorts between adjacent lines or breakage in narrow paths. Some of these visual tests can be facilitated by special test patterns included on the chip. In particular, a series of L-shaped paths (see section 6.1) of various widths and spacings provide quick information about the resolution and cleanliness of the various processing steps.

Special test structures are useful for evaluating the overall process quality or determining process parameters, for example oxide thickness or the conductivity of various levels of interconnects. Some features that have proven useful in our experience are: narrowly spaced interdigited combs to detect bridging between adjacent lines; long, narrow paths to detect breakage; metal paths running across other features such as poly lines to test step coverage; a series connection of many short sections of metal paths and poly or diffusion paths to test for open contact holes.

These and other structures permit one to measure electrical and process parameters. If the geometry of the long paths is known, then the resistivity of the corresponding layer can be derived

from a measurement of the end-to-end resistance. A large MOS capacitor is used for capacitance *vs* gate voltage measurements to determine gate oxide thickness, fixed oxide charge and density of interface states. Some isolated MOS transistors give feedback on threshold voltages and saturation currents in enhancement and depletion mode devices. If the transistors are arranged so that they can be readily connected as inverters, then the characteristic of this all-important building block can also be checked. A circular connection of an odd number of inverter stages forms a ring oscillator, and its the fundamental oscillation frequency yields a good first measurement of the basic pair delay of the inverter stages.

It is often useful to include a set of test patterns on each chip, then if the circuit does not perform as expected the designer can probe the test patterns. The results of those tests give a quick yes/no indication of whether devices on the particular chip are functioning, and hence that the chip was properly fabricated. If the test patterns work correctly, a design or bonding problem may well be at fault. Otherwise the processing is likely to be marginal, and a different chip can be tested.

5.2 Wafer Separation

The wafers, as returned from the fabrication line, are disks of silicon that must be broken into chips that fit into the cavity in a dual-in-line package *(DIP)*. Wafer separation is accomplished in one of two ways: *scribing* or *sawing*.

Scribing is a simple operation similar to glass cutting. The wafer is held on a vacuum table, which is an integral part of the scribing machine, or *scriber*, and a diamond-tipped scribing tool is dragged across the surface within the confines of the scribe lines. Vertical stress cracks are induced where the diamond tip of the scribing tool slides over bare silicon. The pressure that the scribing tool exerts on the silicon is critical, too little results in random breakage in the fracturing operation, while too much produces stress cracks in the horizontal direction. Such cracks cause splintering of the wafer radially from the scribe lines, probably into active circuit elements. After each scribe line in the grid has been "scratched" in this fashion, the wafer (at this point it is still in one piece) is removed from the scriber and fractured into chips. This may be achieved in a number of ways, for example by sandwiching the wafer in some soft material (rubber sheeting, filter paper), supporting it on a foam rubber block, and rolling a cylindrical bar over it. If the wafer was properly scribed the flexing force is concentrated at the scribe lines and the wafer fractures cleanly along them.

Sawing is an alternative to scribing. In this technique a thin saw blade with an edge containing diamond-dust is used to cut approximately 2/3 of the way through the silicon wafer. Again, the wafer is removed from the saw in one piece and fractured by techniques similar to the one outlined above.

Sawing offers several advantages over scribing. The saw can slice anywhere on the wafer, thus no scribe lines are needed. This allows dense packing of projects on a multi-project chip; when the

wafers are returned from fabrication each designer can have a wafer sliced up without regard to the location of other projects on the wafer (i.e. by sacrificing neighboring projects to the saw blade). Most saws can be set up to automatically step across the wafer, making parallel cuts at fixed intervals. This convenient feature provides repeatable die sizing and saves a considerable amount of time, particularly for operators who dice wafers infrequently. Sawing also leaves square edges after fracturing which makes manipulating the chips with tweezers an easy task.

Disadvantages include the necessity of removing the silicon dust (*slurry*) generated in the sawing process — this means an extra cleaning step following wafer separation. Sawing equipment is somewhat more complex and expensive ($12,000-$20,000) than scribers ($2,000-$5,000). Usually there is more maintainance required and more setup overhead involved.

5.3 Chip Packaging

Once the wafers are fractured into chips only *packaging* remains before they are ready to be tested. Packaging encompasses two different operations, *chip attachment* and *wire bonding*. In the first operation the chip is permanently affixed to the IC package; the second involves connecting the aluminum pads on the chip to pads surrounding the package cavity. These package pads are connected through the ceramic to the external pins.

Chip attachment is a straightforward process, especially in a low volume research environment. The chip must be solidly attached to the mounting pad (*header*) in the package cavity. The bond should exhibit low thermal resistance and make good electrical contact with the silicon substrate. Common means of attachment include *solder bonding* (both header and chip must be heated to the melting point of the solder used) and *eutectic bonding* (usually utilizing a gold-silicon alloy, see [Glaser 1977]). By far the most convenient for the researcher is *epoxy bonding*: the backside of the chip or the header is dabbed with a commercially available silver/epoxy mixture and then pressed onto the header. Tweezers suffice for handling the chips. The header and chip are baked at a low temperature for a few hours to cure the epoxy and the assembly is ready for wire bonding.

All three of the manual wire bonding techniques in widespread use require considerable skill on the machine operator's part. *Thermocompression bonding* relies on pressure and heat to produce a strong bond. The header and the chip are maintained at about 350 degrees centigrade. A gold ball (on the end of a fine gold wire) is squashed against the aluminum bonding pad on the chip, forming a bond in a fraction of a second. As the capillary, which guides the wire, is withdrawn from the bonding pad, wire is automatically payed out; the operator maneuvers the capillary over the desired post and the wire is mashed against it, forming the second bond. As the capillary is backed away, the wire is cut by a gas flame that simultaneously forms a gold ball for the next bond. This leaves a short length of wire, from the second bond to the point where the flame made the cut, that must be removed by hand after all wires are attached.

Ultrasonic bonding utilizes aluminum wire and ultrasonic energy to make bonds. The aluminum wire is pressed against the bonding pad and a short burst of ultrasonic energy locally heats the wire/pad interface so that a bond is formed. Similarly, a second bond is made on a post, and the wire is cut, usually by mechanical means. The header may be heated to assist the bonding process.

Ultrasonic bonding offers low materials cost but is less flexible than the thermocompression technique, which allows "daisy-chaining" of connection points. Thermocompression also gives more freedom to choose the angles at which wires leave the bonding pads, enabling some further flexibility which may be needed in a research environment.

Thermosonic bonding effects bonds with a combination of thermal and ultrasonic energy. The header and the capillary are both heated to a somewhat lower temperature than required in thermocompression bonding; ultrasound provides the additional energy to form the bond at the instant of wire/pad contact. This technique has the advantage that the chip and header are cooler than in thermocompression bonding, and so the epoxies used to bond chip to header need not be so heat resistant. At the same time, more wire-positioning freedom is possible than with ultrasonic bonding.

The importance of this wire-positioning freedom should not be overlooked. On a multi-project chip the operator is often faced with less than optimal pad location, and the ability to make bonds at strange angles can mean the difference between being able and not being able to wire bond a particular project. Thus, thermocompression or thermosonic bonding equipment seems most reasonable for those applications.

Chip attachment and wire bonding will doubtless be carried out in a central location, and the packaged chips distributed from there. We have found that the operator of the wire bonding machine can work most efficiently if he or she is allowed to choose the pad/pin mapping as the chip is bonded. For this purpose we provide a number of blank wire bonding maps (see Figure 5.3.1) that the operator fills in as the bonds are made. Once the project is wire bonded, it is packed and distributed with its map. In general, research chips need not be hermetically sealed in their packages, often a piece of tape over the cavity (or no cover at all) will prove adequate. Users should be aware, however, that MOS circuits exhibit very different device characteristics when operated in light (in fact, they may not work correctly).

5.4 Functional Testing

The next step is to functionally test the mounted chip. As mentioned previously, testing really starts at design time. The issues that should be taken into consideration have been discussed in Section 2.7. Here we discuss the actual testing procedure.

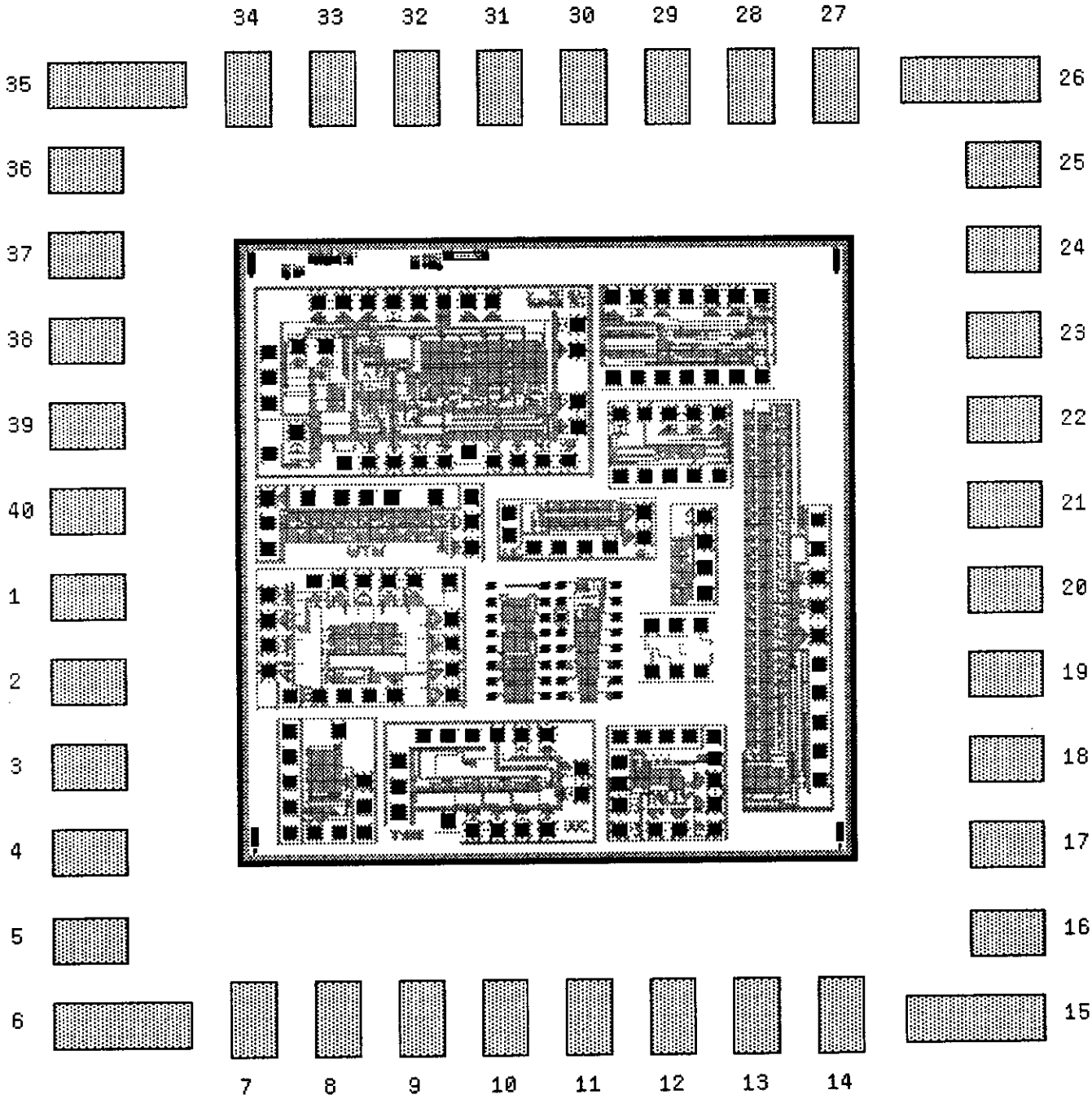


Figure 5.3.1 Wire-Bonding Map

The most burning question to the designer is: *does the chip work?* A preliminary answer to this question should be readily obtainable by plugging the chip into the test set-up that *the designer* should have prepared while waiting for the wafers to be processed. Of course, even if the expected output signals are observed and if it looks like everything is OK, the real testing is only just beginning.

Whether a chip works or not is not just a binary decision. The *quality* of a chip is described by such parameters as maximum operating speed, power consumption and drive capabilities. In addition its *operating margins* must be determined, i.e. what are maximum and minimum supply voltages, maximum clock frequencies and minimum clock pulse widths for which the circuit still works. Even if it is not intended to ever operate under these conditions, the width of these margins will give an indication of how robust the design really is. This ties in with other issues of reliability, for example, is the chip susceptible to external noise pick-up or is the chip subject to an aging process either while sitting on the shelf or during prolonged operation? Even after all these questions have been answered to the designers satisfaction, there may still be some doubts about the *correctness* of a design. Will it really behave properly under *all* allowed conditions — can it be proven? Can the chip be tested exhaustively?

Initial tests may also lead to the agonizing conclusion that *the chip does not work.* The first thought is to blame it on processing. However, since the wafers have been process tested before separation, this is an unlikely excuse. A careful inspection under a microscope often reveals defects which could be responsible for the malfunctioning of the device. Such defects could result from a small local defect in the mask set, from a random defect in fabrication or, perhaps most likely, from mishandling during packaging. Often there are scratches extending several hundred microns from bonding pads into the circuitry.

In most cases there will be at least some signals coming out of the chip, which provide clues about what is going on inside and which can be used to start systematic debugging. Since the logical procedure depends strongly on the particular circuit, debugging is often more an art than a science. If possible a few general rules should be followed. If there are enough intermediate access points, it should be possible to systematically proceed from input to output and to verify the function of subsequent stages until the faulty one is found. It is now that proper partitioning and the inclusion of special testing aids will pay off.

5.5 Simple Test Systems

Except for simple blocks of Boolean logic, typical MOS IC's rarely permit manual testing. Circuit complexity, a large number of inputs, clock signals and outputs, or the multitude of possible states may make manual testing prohibitively time-consuming. Furthermore, if there are any internal nodes which store charge in a dynamic manner, manually exercising such a chip may be too

slow to test its operation. In general it will be necessary to use a computer to exercise the chip (see figure 5.5.1). Properly defined *excitation vectors*, provided by the computer's *output port*, are applied to the inputs of the circuit under test. The *response vector* of the device under test can then be read by the computer's *input port*. This response can then be properly formatted, output in hardcopy, or displayed on a screen in textual or graphical form. Alternatively, it could be compared to a stored correct response, with the computer programmed to respond with an error message upon detecting deviation from the expected pattern. In any case, the test vectors must be carefully selected if they are to supply relevant information about the chip.

Typically, only a small amount of hardware is required between the computer and the device under test (for example, see [Mathews 1979]). A minimal system could consist of a general purpose test board with a zero insertion force socket, a tristate driver and read buffer connected to each pin, a couple of latches to store the state of the input and/or output variable at each pin, and a communication interface to the computer. For simple chips, a serial data line, transmitting individual characters for each change of the state of an input pin or for each read request of some outputs, may be sufficient. Obviously the maximum rate at which a chip in such a set-up can be exercised is limited by the bandwidth of the serial link and the number of pins that need to be changed in each phase. This approach has the advantage that a minimal amount of additional hardware is needed. Such a software-based testing system, however, might not be fast enough to capture some quickly changing response vectors, or it may have trouble exercising a complex device which may require some minimum clock rates for proper operation. Testing the maximum operating speed of an IC also requires input rates which often can not be provided by a simple software system.

More performance can be obtained with a parallel link to the computer. If the excitation vector is wider than the typical 8 or 16 bit link to the host computer, the vector may be transmitted in several pieces and assembled in a set of registers on the interface board. Even higher I/O vector speeds can be obtained by moving more functions from software into hardware. An improved tester could use semiconductor memory to store the excitation and response vectors of a whole test sequence. A hardware counter is used to step the memory through the required words. The captured response vectors can later be analyzed at a slower rate. If two memory banks are used each for input and output vectors, the device can be kept running in a continuous test loop, while the other input bank is reloaded with the instructions for a new test cycle or while the second output bank is read by the computer.

The tester is now a *peripheral* device to the host computer (see figure 5.5.2). The advantage of ever more self-contained testing hardware is two-fold; it allows higher testing speeds and frees the computer to perform other tasks while testing is in progress. The ultimate step is to provide a stand-alone processor for testing, flexible enough to test any conceivable digital IC. The excitation vector for the device under test can be understood as a set of *control words* emanating from a computer's *control unit*, with the result vector out of the device acting as a *condition vector* to this

Figure 5.5.1
Software approach to IC testing. A computer is dedicated to exciting the device under test (DUT) and collecting the response. The disadvantage of this method is that it is slow and wasteful of computer time. Note that the computer is directly connected to the device under test.

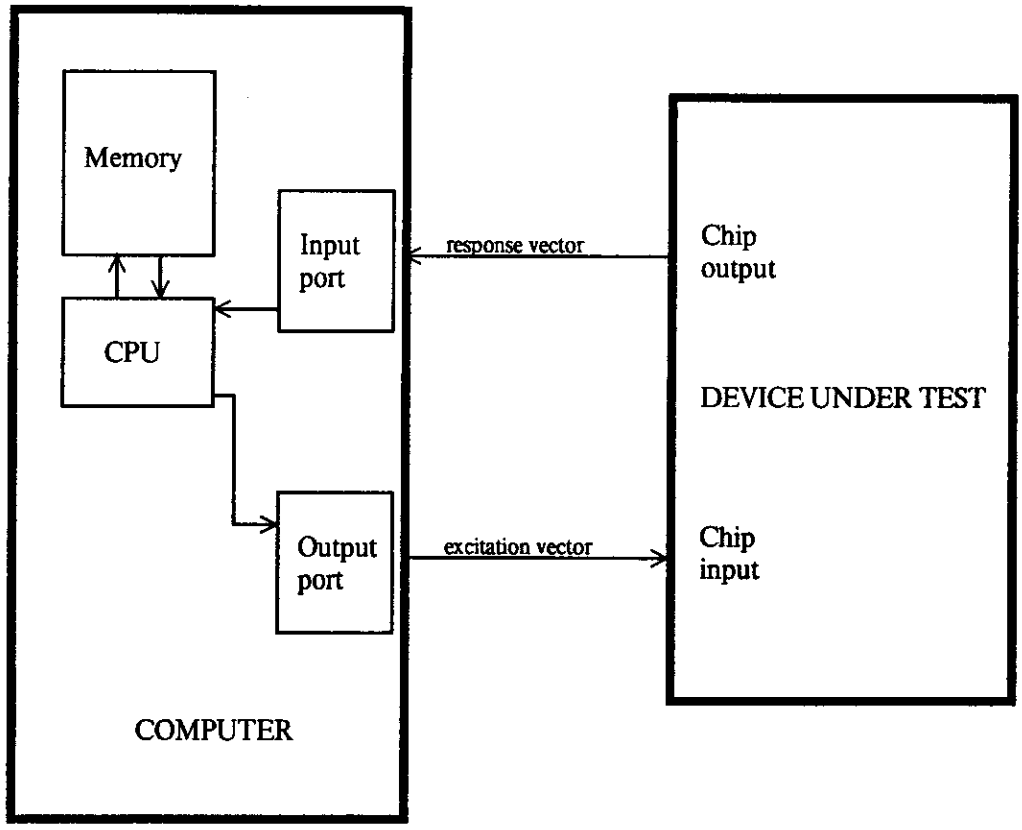
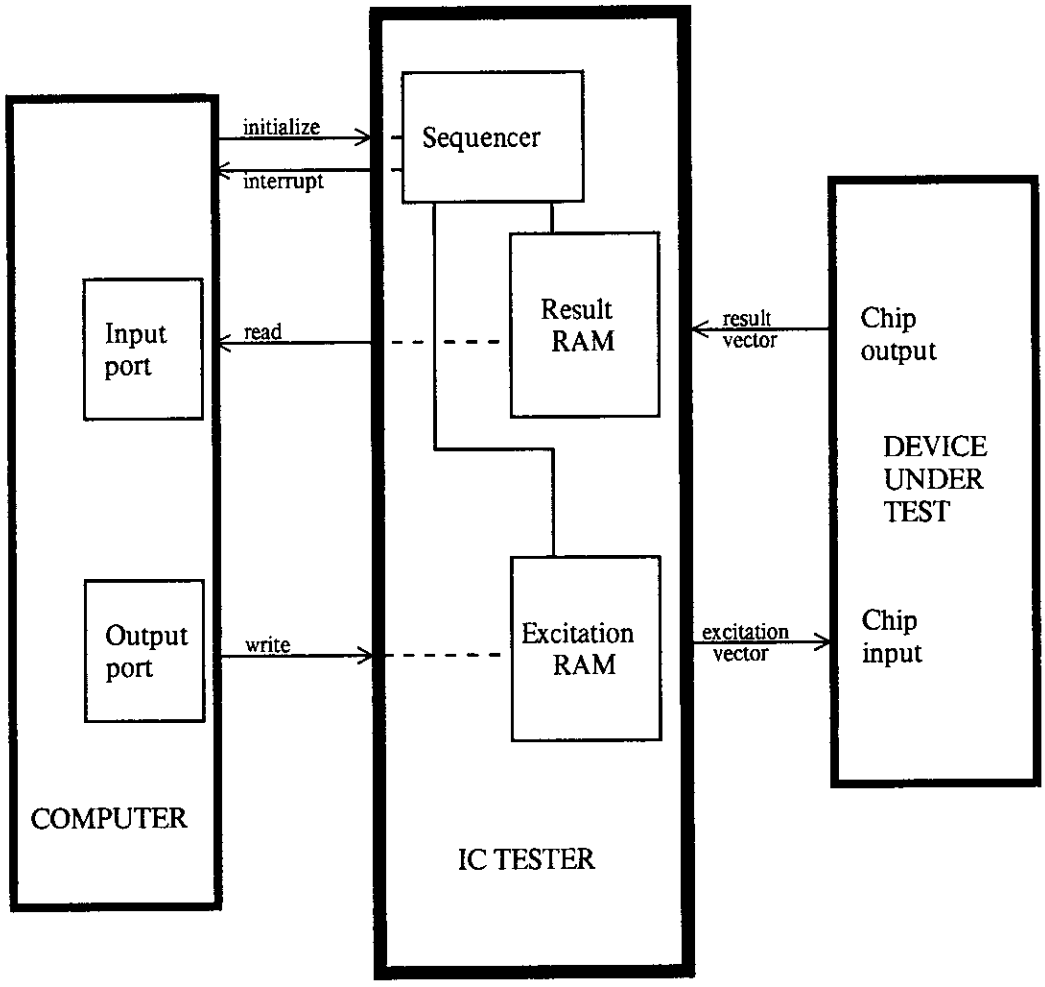


Figure 5.5.2
Hardware approach to IC testing. The computer initializes an IC Tester that is connected as a peripheral device. The sequencer (counter) steps both RAMS, sending an excitation vector and collecting the result vector. When the test is over, the sequencer interrupts the computer. This method is fast and requires little CPU time. Note that the computer is no longer directly connected to the DUT.



control unit. The tester thus takes the form of a *microprogrammed* controller. A tester based on this principle can exercise very complex devices due to its inherent ability to make logical decisions based on some of the results. When the test is concluded, relevant results are as before stored in a result RAM, and the host computer is signaled to fetch them. This kind of a tester can be adapted to new tasks by changing the *microcode* (see figure 5.5.3); it may even do suitable branching dependent on the outcome of a few preliminary tests.

Almost any computer or microcomputer can be used as the host. The only requirement is that it have an accessible I/O port. The control unit for the tester could be built using one of the fast bipolar *bit-slice* microprocessors.

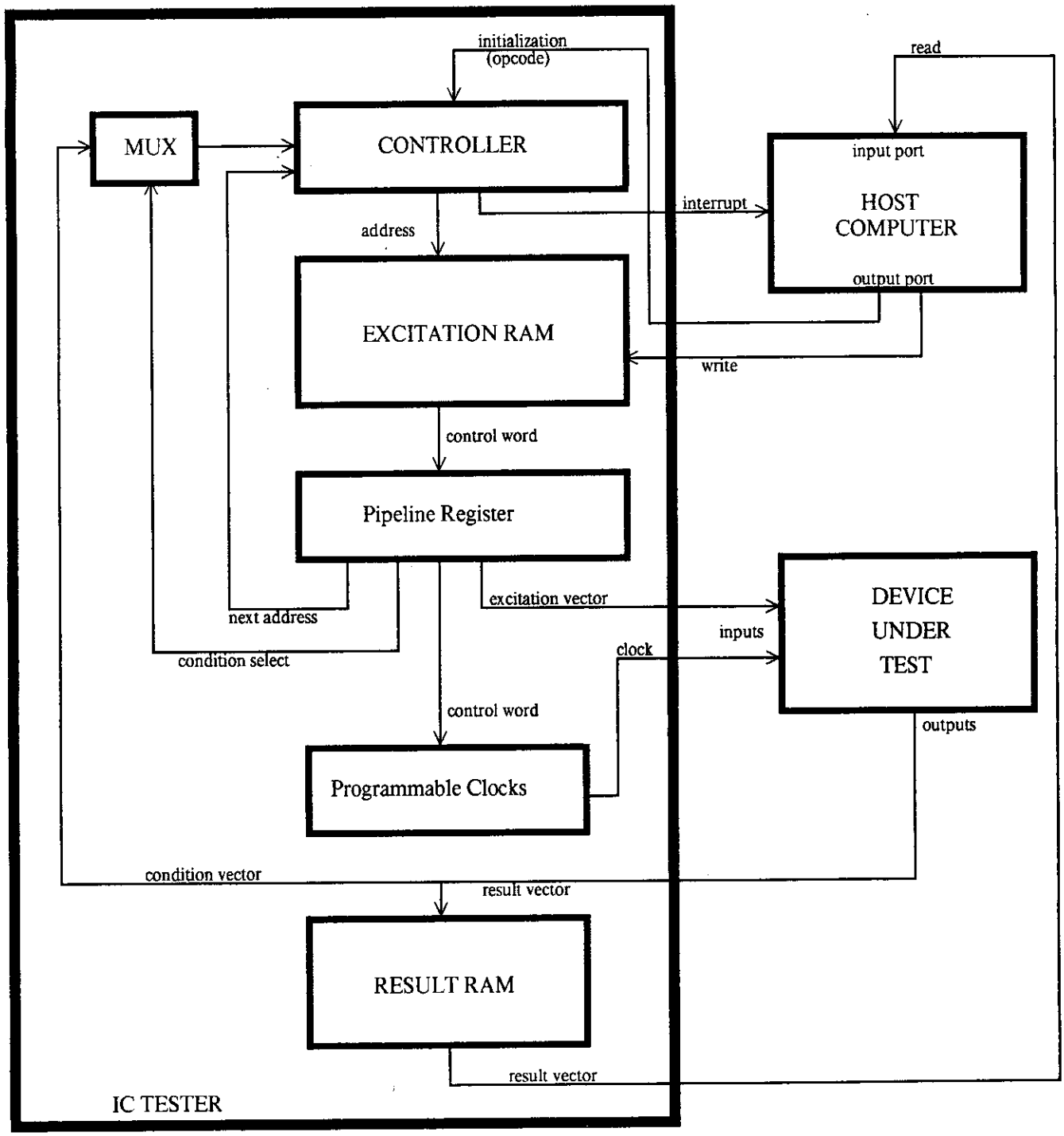
5.6 A Concluding Remark

It should be emphasized that preparing for the day when the wafers come back from the fab line may be as large and complicated a task as the original design. The proper custom made interface board between the chip and the test system has to be built and the test routines have to be written. In preparing for testing, the designer should keep in mind the possibility that the chip does not work at all and plan a strategy to deal with this case. Debugging and testing is the responsibility of the designer and should be kept in mind from the early stages of the design process to the day of the delivery of a finished product to a customer. The availability of quick turnaround IC implementation could permit a new and more experimental way of designing and debugging integrated systems, in the same style that programmers now use to build large software systems. It is clear that this iterative design loop must be closed by the IC designer, who must provide the hooks to extract all relevant information from the silicon chip.

References

[Mathews 1979]
R. Mathews, "A Minimal Tester", Stanford CS Technical Report, November 1979.

Figure 5.5.3
Block diagram of a Microprogrammed IC Tester. This approach allows maximum flexibility by being programmable. The computer initializes the controller which outputs the address of the first microinstruction to the microprogram RAM. The microprogram RAM supplies the next address and enables the controller to sequence through various testing microsubroutines.



CHAPTER 6
AN EXAMPLE STARTING FRAME AND PROJECT CHIP

The starting frame and test patterns described in sections 6.1 and 6.2, respectively, were implemented on a multi-project chip during the summer of 1978. That chip contained ten projects, ranging from process test structures to novel arithmetic and memory circuits. Figure 6.1 shows the overall layout and one mask layer of the chip. The chip includes a set of alignment marks and line-width testers laid out by Bob Hon and Dick Lyon and a general process test chip laid out by Rick Davies.

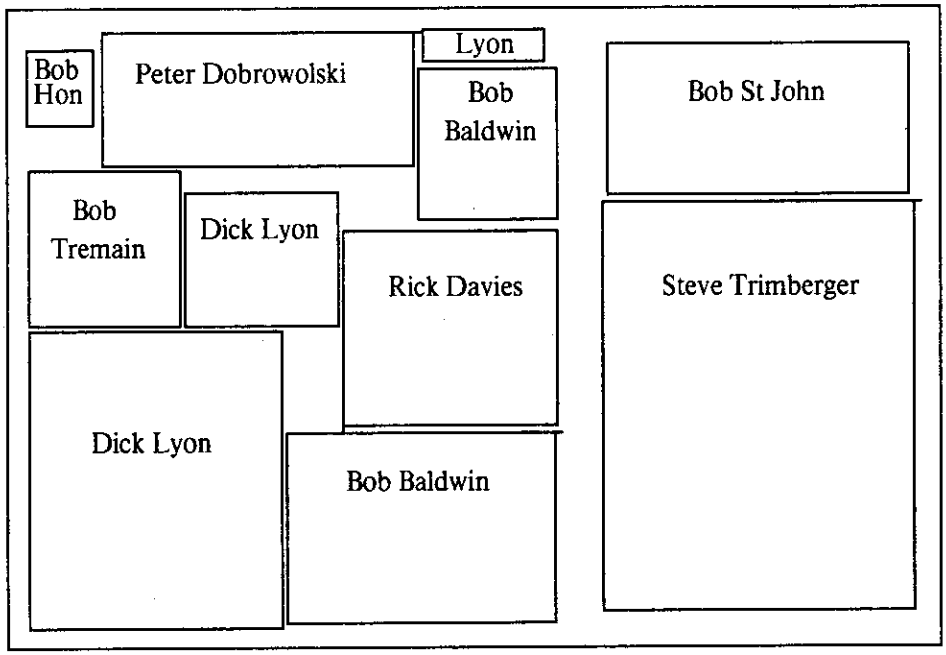
In this chapter the features of the starting frame are presented first, followed by a discussion of the general test chip. Finally in section 6.3, Jim Cherry, a graduate student at MIT, details a project that he designed for a course given there in the fall of 1978.

6.1 The PARC Starting Frame

The multi-project chip is divided into two parts separated by one internal scribe line so that both parts are small enough to fit inside the cavity of a 40 pin DIP. Further, the layout is enclosed by exterior scribe lines placed around the periphery. The exterior lines differ from the interior lines in that the former are missing the outside "shoulder" (Figure 6.1.1); the exterior lines of one pattern are completed by the overlap with the exterior line of the next chip. The scribe lines are designed to provide direct access to the Si substrate for the scribe tool during wafer separation (see section 5.2).

The alignment marks (Figure 6.1.3) were intended to unambiguously indicate which layers are to be aligned relative to each other. The marks consist of a number of "squares" and "fortresses". A square mark is placed on those layers that will serve as reference layers for masks in following fab steps. Each square has a corresponding fortress, located on a different mask, which will be aligned over it during the appropriate step (see Figure 6.1.2). Each layer includes a large rectangle around all of the alignment marks to help the operator to locate them and to insure that the sequence is not shifted. The features are lines rather than areas, permitting the operator to align edges relative to one another. This makes the marks usable for clear as well as dark field working plates.

The fortress/square pairs are used in a left to right progression; a digit (omitted in the figures below for clarity) is placed in each fortress to indicate when it is to be used. A fortress is always aligned over a square and there is never more than one fortress per mask. The alignment sequence has the depletion mode implant, buried contacts (when used), and the polysilicon layer all aligned relative to the diffusion layer. The contact cuts are aligned relative to the polysilicon since there



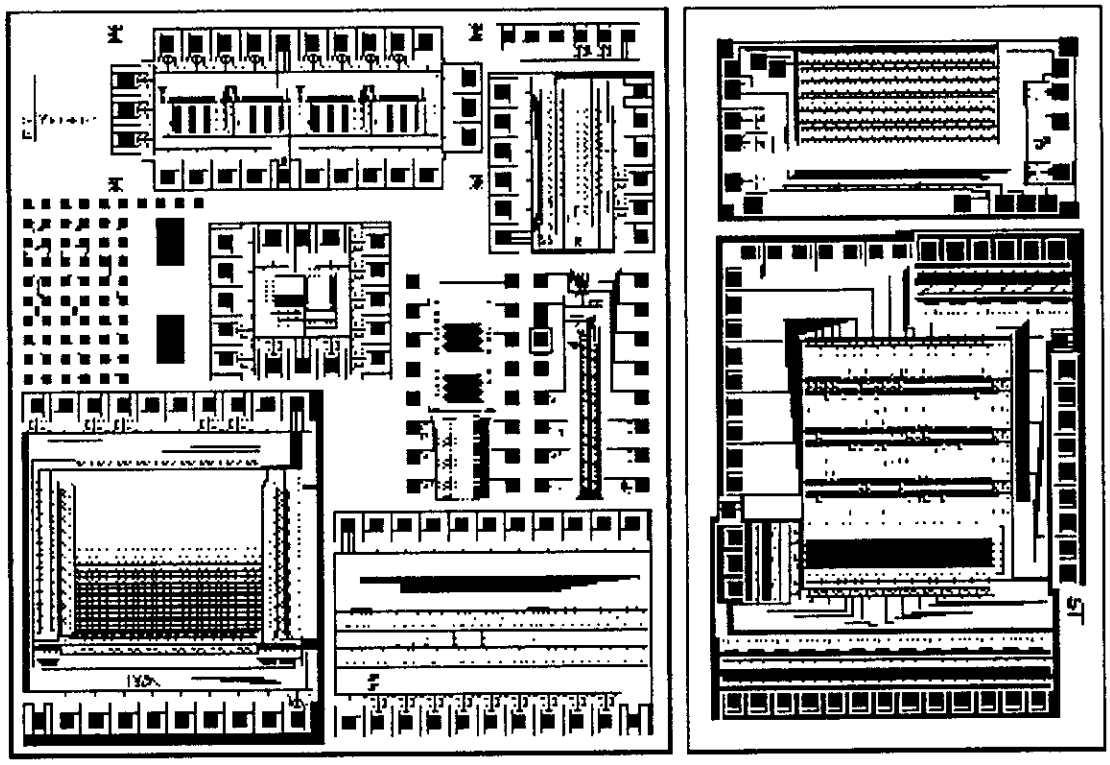
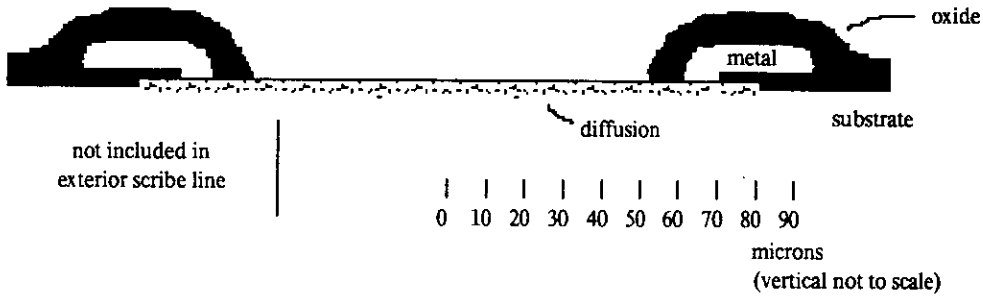


Figure 6.1 The Summer 1978 PARC Multi-Project Chip

Figure 6.1.1 Scribe Line Profile



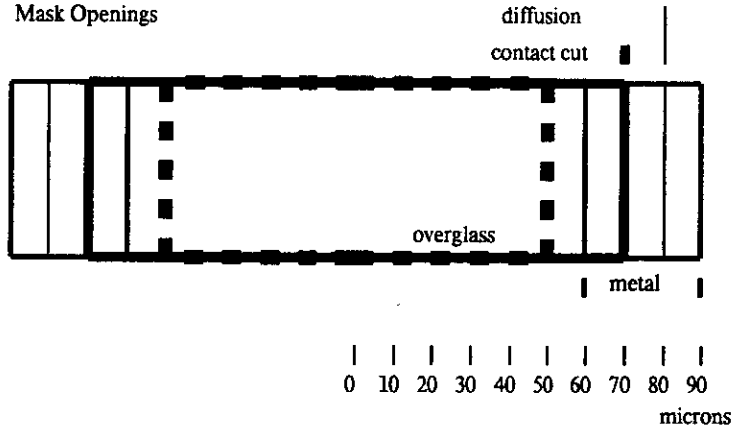
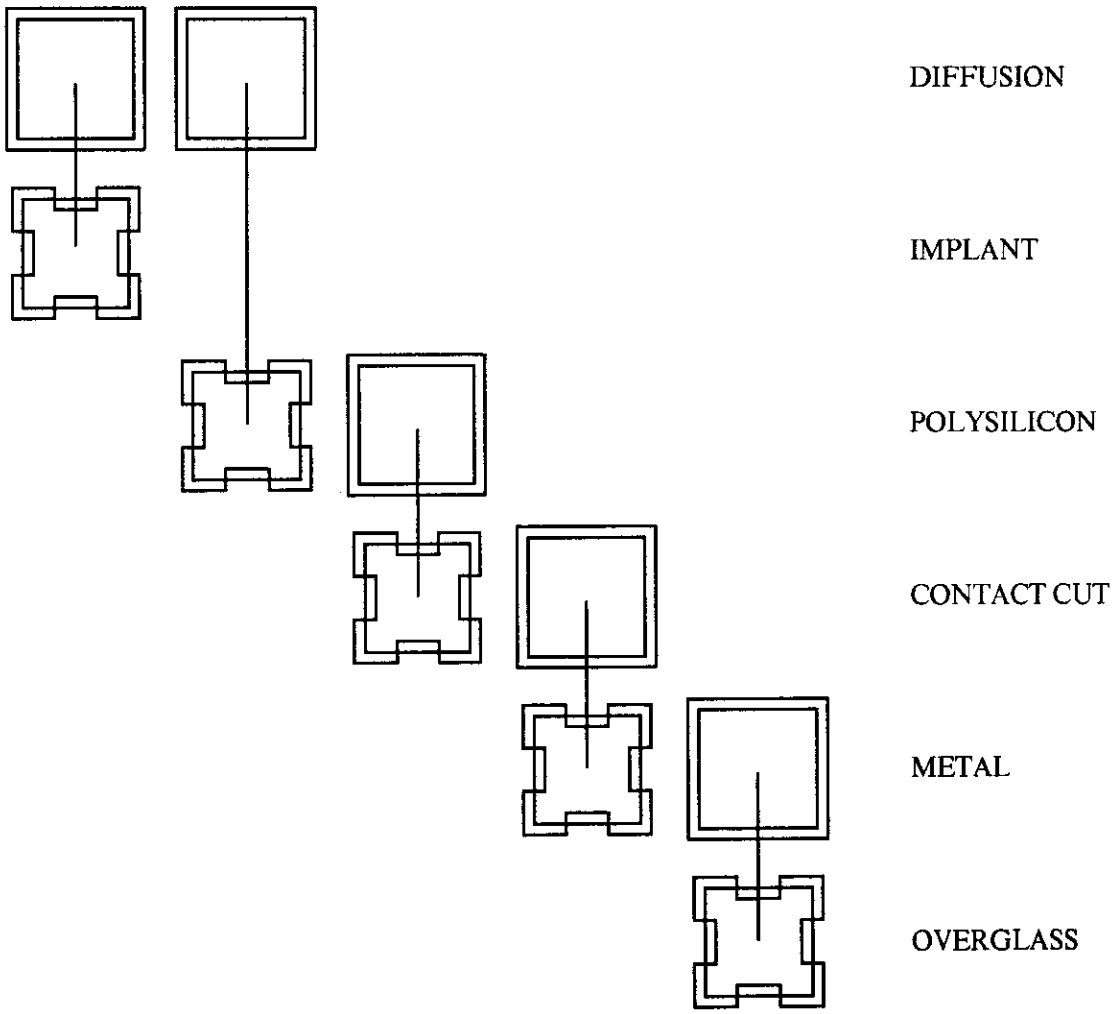
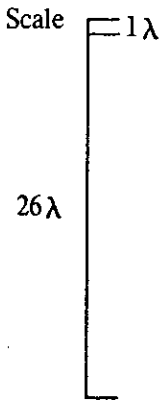


Figure 6.1.2 Alignment Marks for Mask Layers





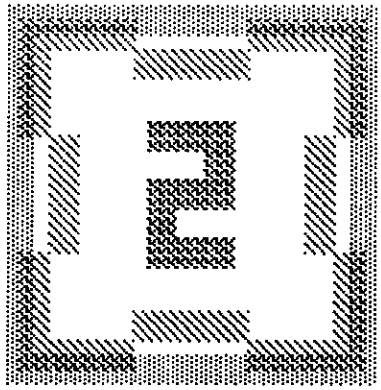


Figure 6.1.3 Alignment Mark

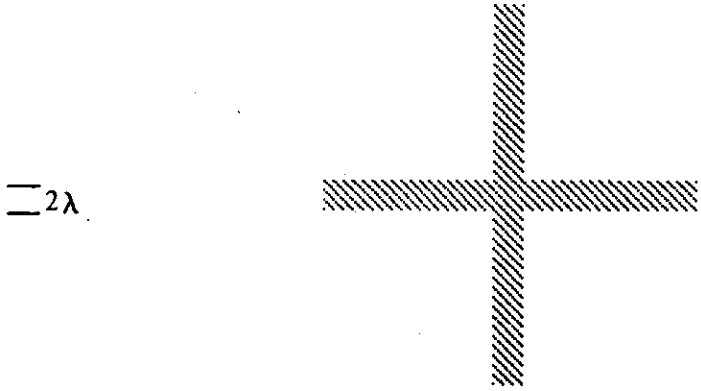


Figure 6.1.4 Critical Dimension Cross

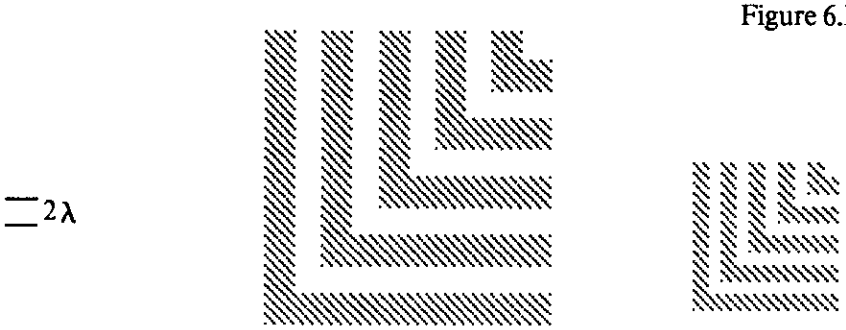
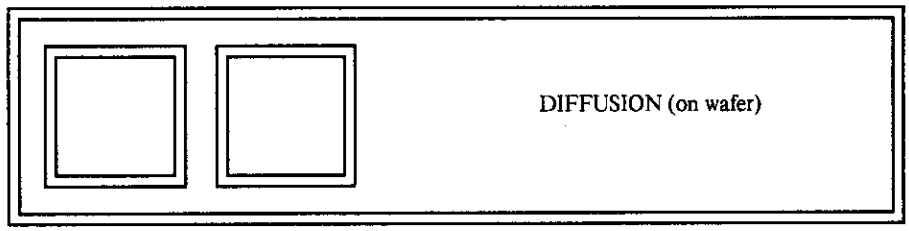
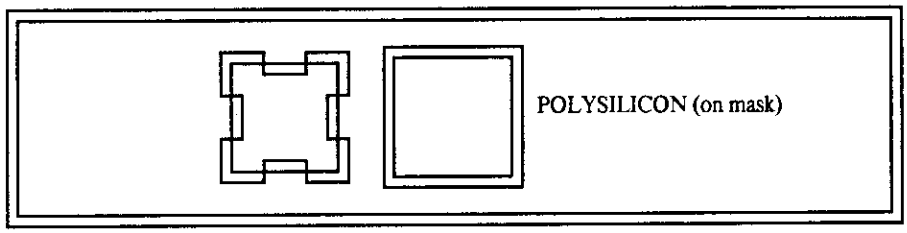


Figure 6.1.5 Etch Test Patterns

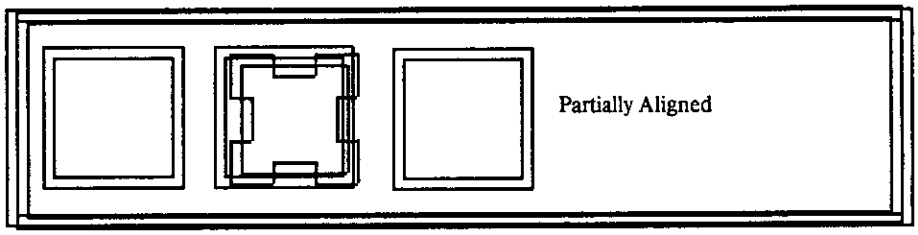
appears to be more tolerance to misalignment between contact cuts and diffusion. The metal aligns to the contact cuts and the overglass to the metal layer. The following illustration represents the way an operator might align the mask for the polysilicon layer to the wafer. The pattern on the partially processed wafer is:



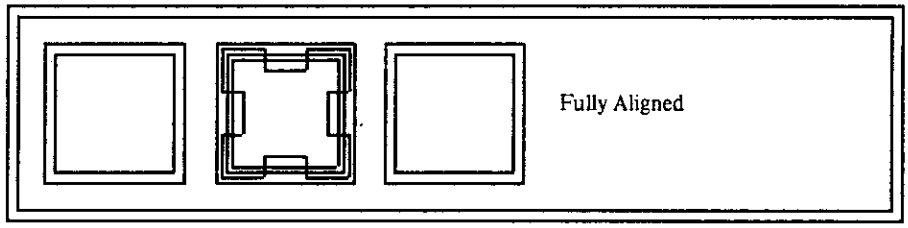
The operator must now align the polysilicon mask, which contains one fortress and one square.



The diffusion and depletion implant steps are already complete, thus the first fortress/square pair has been used. The operator lines up the second pair using the gross alignment mark for guidance. The following diagram shows the mask in partial alignment with the features in place on the wafer.



Final corrections are made using the fortress/square pair.



The third square will be used to align the contact cut layer; a fourth square is placed on the wafer during the contact cut step and will be used to align the metal layer.

The critical dimension marks are simple crosses made of lines (Figure 6.1.4). The line widths vary from layer to layer, and are typical of the feature dimensions found on the particular layer.

A set of features used to monitor the quality of the working plates and fabrication process was also included. This *etch test pattern* (Figure 6.1.5) consists of a set of nested "L"'s with the same spacing between L's as the width of the feature; two different sizes were included on each level to check the quality of the mask and the quality of the photolithographic process.

Measurements on each mask layer provide a check on the dimensional correctness of the working plates while measurements on the wafer are used to verify that the fab line performed as anticipated (e.g. that lines over- or under-etched as expected).

Appendix A contains a copy of the information sent to the mask house.

6.2 Test Patterns

[contributed by Rick Davies, Xerox PARC]

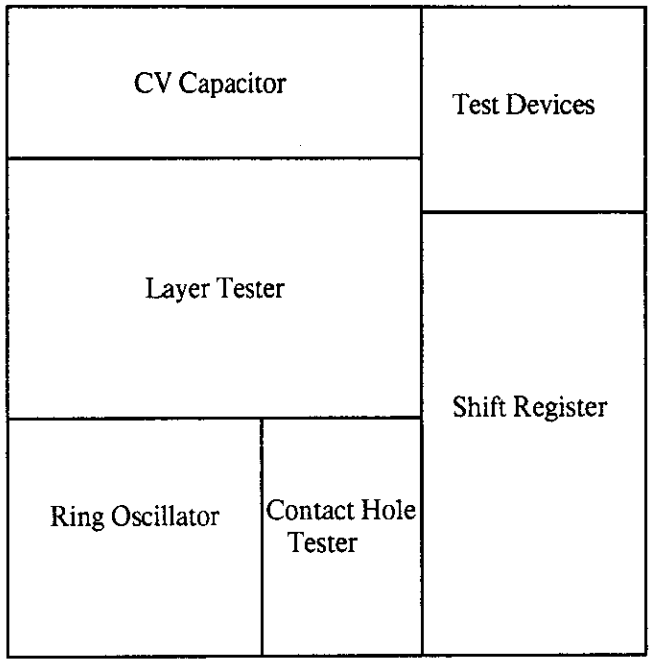
The starting frame contains a number of simple test structures to answer the following two questions:

> Was the wafer properly processed? Specifically are all the layers properly patterned, are gate oxide and deposited oxide films of acceptable dielectric integrity, are contact holes properly opened, etc.?
>
> What are the first-order device and circuit performance characteristics such as transistor threshold voltages, extent of short- or narrow-channel effects [Dennard 1974, Wang 1978], polysilicon sheet resistivity, and inverter propagation delay obtainable with the process?

The test structures described here are general enough so that it is assumed that they will prove useful to most participants in a multi-project chip. This should not deter any designer from adding his own special test structures. It may be desirable in the future to add additional patterns to the common test structure, to test such parameters as the quality of the buried contacts between

diffusion and polysilicon or the limits of wafer processing (e.g. At what spacing do metal lines begin to show bridging?).

The test pattern consists of several separate regions which are described below. It occupies 2 mm x 2 mm and has this general layout:



6.2.1 Layer Tester (Figure 6.2.1)

This is a long serpentine metallization path that runs between two interdigitated metal combs and lies over a serpentine of polysilicon and active transistor area. It tests the following features.

Metal bridging. There is an 8400λ periphery of minimum-spaced (3λ) metal lines between the serpentine and combs. Conductance between the serpentine and either comb indicates bridging, which could be caused by improperly patterning the photoresist or incompletely etching the aluminum.

Metal step coverage. The 4λ-wide serpentine passes 266 times over a 2λ-wide region of polysilicon, over gate oxide, between 2λ-wide source-drain diffusions. This should provide a good indication of step-coverage quality (assuming that the above bridging test has been passed), measured as a low end-to-end impedance (<100Ω). Metal running over diffusion and polysilicon is the worst-case condition for metal step coverage; the limited solid angle provided by the evaporation source can make it difficult to deposit aluminum on the vertical features in small-geometry devices.

Gate-oxide dielectric integrity. A polysilicon electrode of approximately $11{,}000\lambda^2$ over gate oxide provides a test for pinholes and shorts between gates and transistor-channel area. This is equivalent to the gate area of about 1000 transistors of typical size. To pass this test there must not be measurable conductance between the polysilicon and the diffusion.

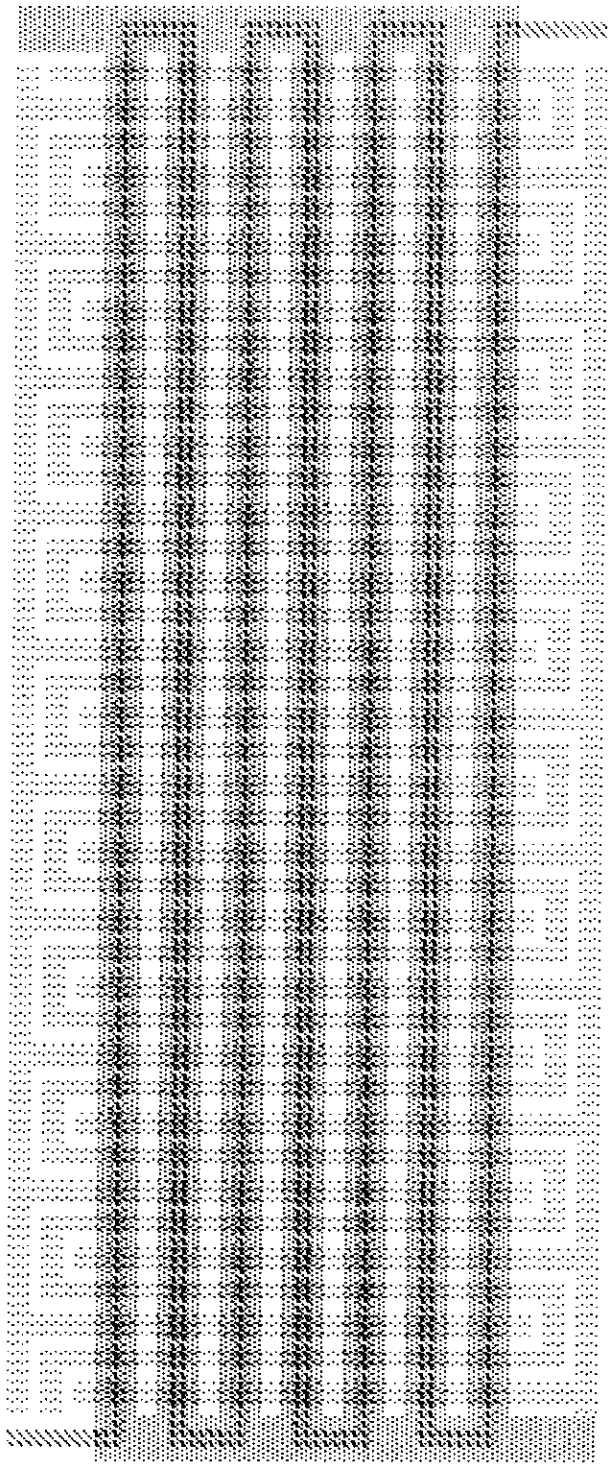


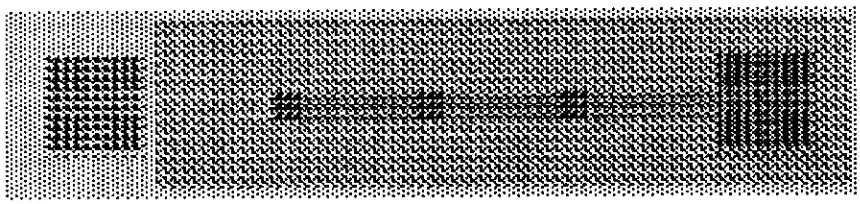
Figure 6.2.1. Layer Tester

Deposited-oxide dielectric integrity. The presence of approximately 11,000λ^2 of metal over active area (polysilicon gate or source-drain diffusion) provides a test for pinholes and shorts to the metallization; this could result from improper annealing of the metal causing spiking through the deposited oxide layer. No conductance should be observed between the metal and either diffusion or polysilicon.

Polysilicon and aluminum sheet resistivity. Although more accurate resistivity measurement can be made using a Van der Pauw structure (separate forced-current and sensed-voltage terminal pairs), polysilicon and aluminum serpentines provide a quick estimate by inspection of the end-to-end resistance. About 700 squares are present in either level.

6.2.2 CV Capacitor.

A 68λ x 289λ MOS capacitor with a diffusion guard-ring is provided for analysis of the process parameters Q_{SS} (density of fixed charges at the oxide-silicon interface), N_{SS} (density of trapping states at the interface) and the gate oxide thickness [Grove 1967]. A minor amount of wafer preparation may be required to form a suitable ohmic backside substrate contact for reliable measurements.



Measurements on this structure would allow separate determination of the implant dose and interface characteristics components of the threshold voltages of the enhancement and depletion NMOS transistors. This structure may also be used to test gate oxide dielectric integrity; 20,000 λ^2 are present.

6.2.3 Contact Hole Testers

The following two contact-hole tests are incorporated on this test pattern (Figure 6.2.3).

A series connection of 270 metal/polysilicon contacts (2λ x 2λ) tests for failure to make electrical contact between metallization and the underlying layer. A measured resistance significantly above the expected impedance corresponding to the parasitic 135 squares of connecting polysilicon indicates poor quality ohmic contacts. This could be caused by improper imaging of the pattern in the photoresist (in particular a scum residue might have been left in the bottom of a hole), improper etching of the oxide, or by metal breakage around the rim of the etched contact hole.

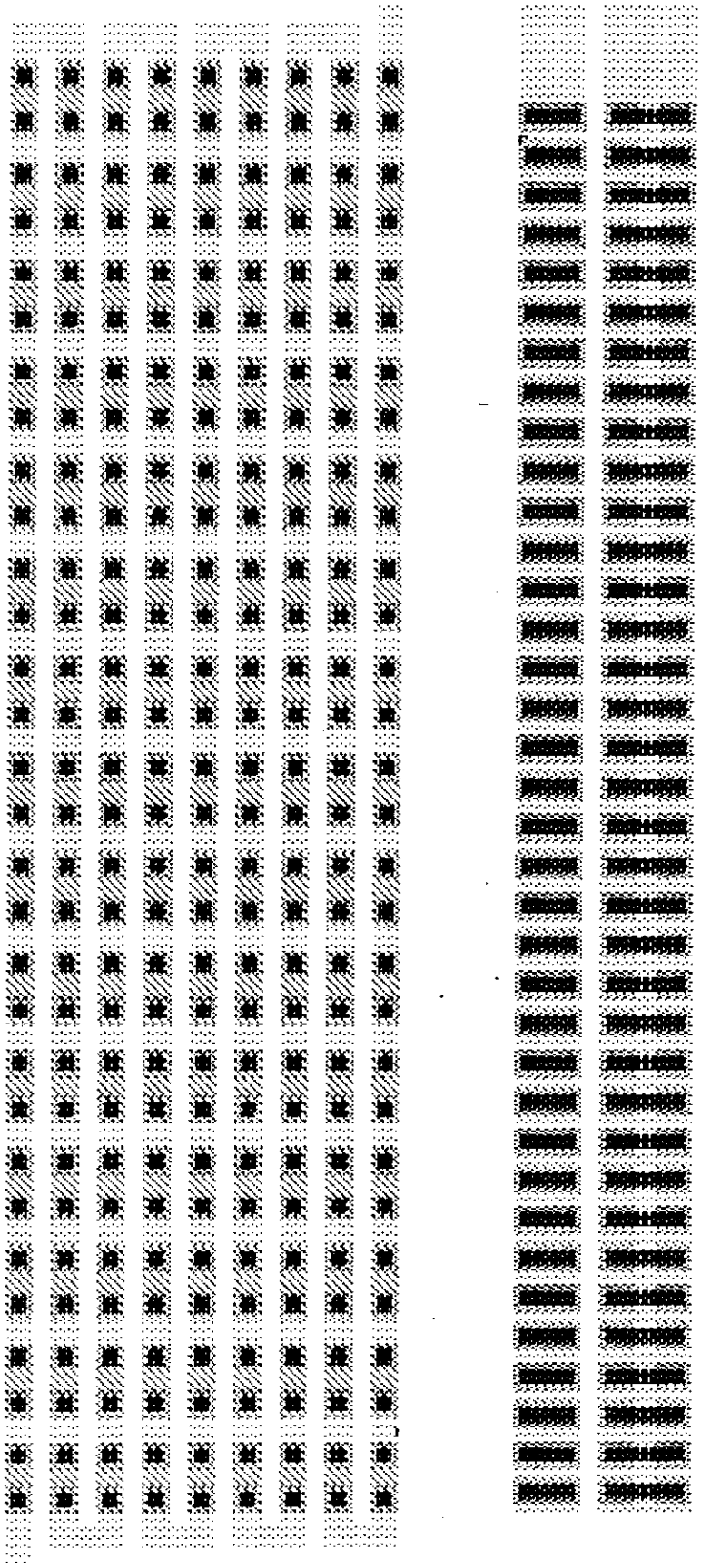


Figure 6.2.3
Contact Hole Testers

A special tester consisting of 2 columns of 36 metal/diffusion contacts compares contact holes that are properly centered over a diffused area and others that overlap the field oxide region. The latter is used to test whether one could overlap contact holes onto the field oxide, in order to save the area otherwise consumed by alignment tolerance. Because the phosphorus-doped SiO_2 (sometimes called *P-glass*, see section 3.3 for a description of the fabrication process) deposited before contact hole definition etches much faster than does the thermally grown field oxide, the contact holes should open before the field region is etched through.

The two columns have identical bottom-wall diffusion area, diffusion periphery, and contact hole area over diffusion; the right one differs from the left only by the incorporation of a strip of field oxide in the middle of the contact hole. If the two columns reveal the same leakage characteristics to the substrate, then this overlap technique is probably acceptable.

6.2.4 Discrete devices

Four enhancement and four depletion mode transistors are provided for dc testing (Figure 6.2.4). They are organized as four inverters for convenient transfer curve analysis, with uncommitted gates for the depletion-mode transistors to allow full testing of those devices. To minimize the number of bonding pads, the enhancement-gates are shared, as are the depletion-gates; all enhancement-sources and all depletion-drains are also shared. The four inverters are:

2λ-length/4λ-width enhancement NMOS with 4/2 depletion NMOS, forming a standard 4:1 inverter.

2/4 enhancement and slightly narrowed 3.3/2.5 depletion load device. One expects a higher threshold in the narrowed channel because the channel potential is raised by the increased influence of edge effects. The use of scaling [Wang 1978] (which involves altering the fabrication process) would permit this smaller layout without disturbing the dc characteristics.

2.5/3.3 enhancement and 4/2 depletion load devices. Short-channel effects should cause a lowering of the threshold voltage and produce increased output conduction in the enhancement device [Dennard 1974].

20/40 enhancement and 40/20 depletion load devices. These devices should permit one to check device characteristics with little interference from peripheral effects.

Two transistors with closed layouts — one with a metal and the other with a polysilicon gate — on thick field oxide, are provided to test for isolation-region channeling or other parasitic leakage. A threshold voltage above about 25v should exist on each device. In both cases the transistor gate electrode overlaps the source and drain regions. The poly-gate structure makes gate-oxide devices at source and drain that are in series with the field-region under test.

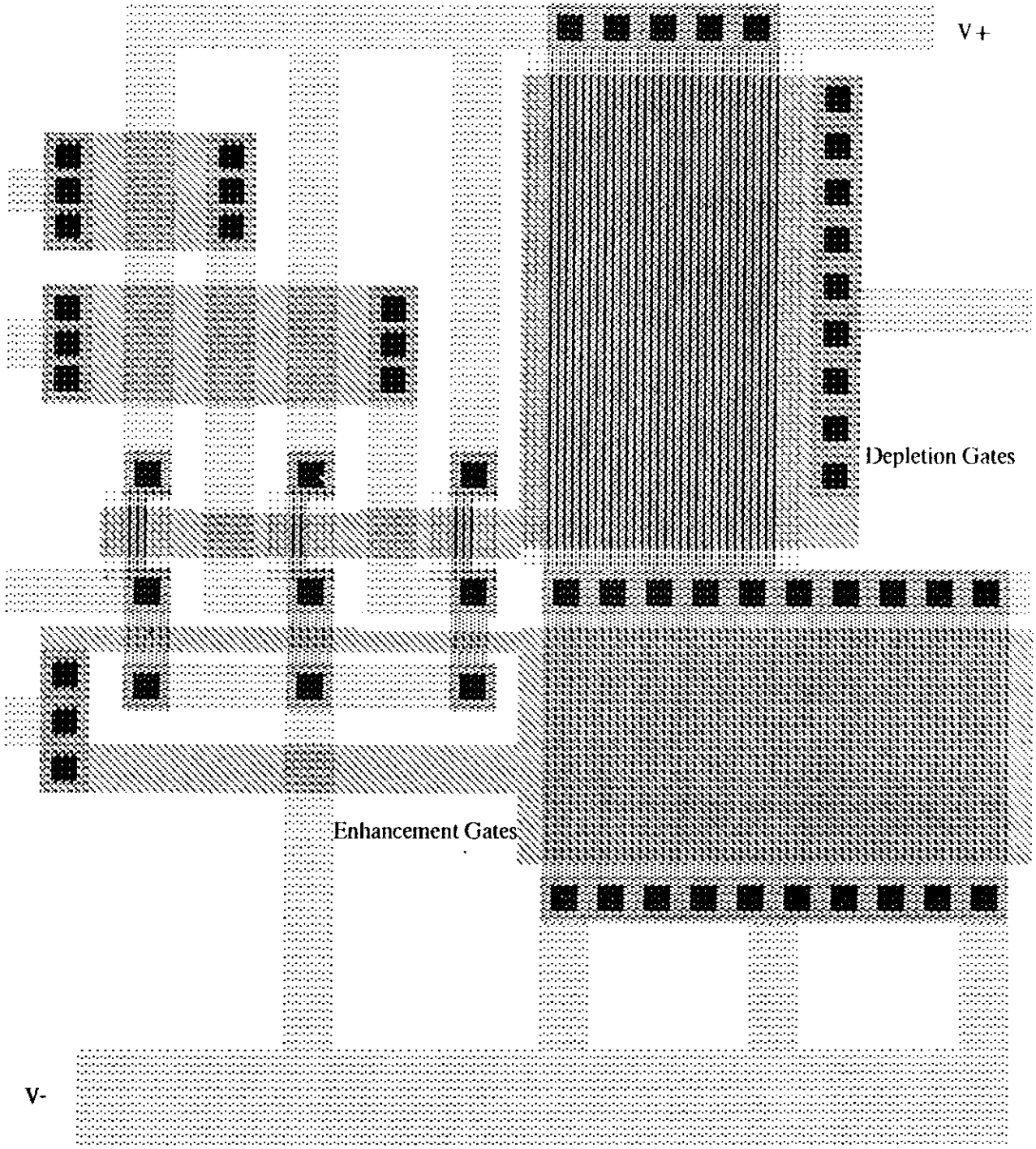


Figure 6.2.4 Discrete Devices

6.2.5 Ring Oscillator

This structure tests ac device performance in a probe environment without hindrance from the inherent parasitic capacitances. Because it is an actual circuit, it should provide more reliable and directly usable information than that derived from separate detailed ac device measurements and circuit simulation techniques. The ring oscillator is 25 stages long to provide a low frequency output signal. The average propagation delay is one half of one twenty-fifth of the inverse of the natural loop oscillation frequency. Inverters with $2\lambda/4\lambda$ enhancement drivers and $4\lambda/2\lambda$ depletion loads are used; a buffer/inverter taps the loop, giving one of the 25 loop inverters a fanout of two. The buffer in turn feeds an output transistor with L/W of 1/20; this may be used either as a common-source output driver or as a source-follower.

6.2.6 Shift Register

This 33-stage circuit is similar to the above ring oscillator, with the addition of a passgate in front of each inverter (they are $2\lambda/8\lambda$ enhancement drivers and $4\lambda/2\lambda$ depletion loads; the passgates are $2\lambda/2\lambda$ devices). The passgates are bussed in two phases that alternate between inverters (16 passgates per phase). The 33rd passgate is brought out to a separate bonding pad so that one can open the shift register loop. The complement of this signal is applied to another passgate that connects the shift register to a separate, inverting input buffer.

With both passgate phases and the control passgate line high, the circuit implements a 33-stage ring oscillator using passgate signal transmission. The measured average propagation delay may be compared to that of the simple ring oscillator (described in section 6.2.5).

With the control passgate line low and alternate clocking of the passgate phases, the circuit acts as a shift register which may be loaded with an arbitrary bit pattern. Raising the control passgate line while continuing two-phase clocking (at a rate below that of the loop self-oscillation frequency) forms a recirculating shift register.

6.3 Example Project: A Transformational Memory Array

[contributed by James J. Cherry, MIT]

6.3.1 Background and Abstract

In this section the design of an LSI chip will be described from its inception through the selection of the architectural structure and on through final layout, to illustrate the integrated nature of the process. The selected example is as one of the projects produced during the 1978 VLSI design course taught by Lynn Conway at MIT. It is a special purpose NMOS memory array for use in an object oriented graphics display. The memory is an 8x8-bit array which may be non-destructively accessed in serial-raster scan format. This bit map can be read in its original orientation, mirrored about the X or Y axes, and/or rotated a multiple of 90 degrees.

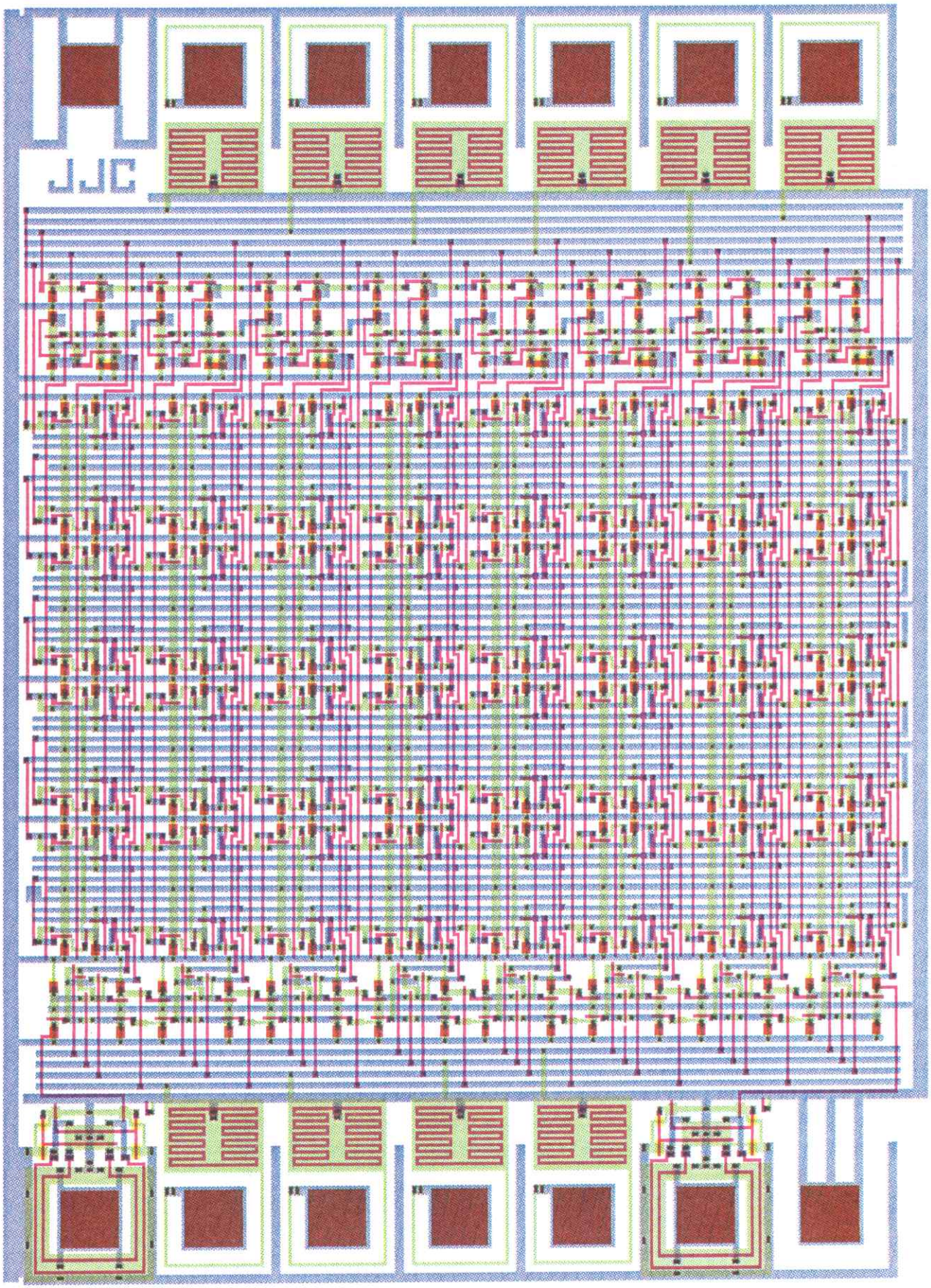
6.3.2 Project Context

In many video graphics applications, a display may have a variety of logically distinct objects. It may be desirable to have the capability of placing each object at an arbitrary orientation and position on the screen. An example is a game in which space ships or cars move under user control. In addition to moving across the screen, it is desirable to have the vehicle point in the direction it is moving.

While this may be accomplished using a bit map storing the pixels on the entire screen, operations such as those mentioned above require a lot of computation. Even the translation of an object is difficult, since an object will most likely change its position relative to the word boundaries when moved. To perform mirror images or 90 degree rotations, requires an even larger number of bit manipulations.

In the context of logically distinct objects, the hardware could be made to follow the structure of the objects to be displayed. In other words, create hardware that is specialized to display a single object, where each module is as intelligent as hardware limits permit. For instance, the module may know the coordinates of its upper right hand corner. As the display is raster scanned the modules act as demons, waiting for the addresses which correspond to its region to send their content to the display. This makes the translation of an object a simple matter for the processor. It merely updates the corner register's contents. The outputs from a number of modules can then be ORed together to form a composite display. A fixed pattern (background) may also simultaneously appear by ORing in its contribution.

Associated with each module may be a variety of attributes. Examples include transformations performed on the pixel matrix such as mirroring in X or Y and rotation. The color of the object may also be considered an attribute. A collection of these modules allow a number of objects to be simultaneously displayed. This frees the processor for higher level tasks.


JJC

6.3.3 Project Scope

The above module can only be useful if a several of them (one for each separate object to be displayed) are available. To implement even the simplest hardware module, capable of displaying a single object in its original orientation, requires on the order of fifteen TTL packages. This renders the approach quite uneconomical when implemented with off-the-shelf SSI/MSI integrated circuits. Only, if the module itself is implemented in a single IC, the approach becomes reasonable. To be truly useful the modules should be smart enough to perform localized operations on the data itself. Mirroring and 90 degree rotations are almost required transformations. Rotations through arbitrary angles do not have simple mappings from the original to the transformed bit map, and are thus very difficult to implement.

The chip to be described is an (8x8)-bit memory array that is read in raster scan format suitable for use with a video monitor. The bit map array can be read in its original orientation, mirrored about the X or Y axes, and/or rotated a multiple of 90 degrees. I have decided not to implement the coordinate detection logic, primarily because it is both interface dependent and thus limits the scope of the matrix manipulator's usefulness. The final memory has more uses than just in the context of a graphics display. One example is to perform these operations on a matrix of N-bit numbers; this requires stacking the modules N deep.

6.3.4 Algorithm Description

Figure 6.3.1 shows a block diagram of the sub-system that was implemented as a project. A square array of eight by eight one bit data cells is arranged in the center. The data cells are serially loaded with a bit map corresponding to the object to be read in a given orientation. Along the top of the array are a pair of shift registers which can propagate a *select* pulse along the top edge of the matrix in either the left or right direction. Each *select* line selects a vertical slice (column) of the bit matrix to be read. By propagating the select pulse from right to left or left to right, the columns of the array are read in their original orientation or their horizontal mirror image. The selected column is then loaded into a pair of shift registers at the bottom of the array which can also be shifted in both directions. These output shift registers are used to mirror the selected column about the horizontal axis. The serial output of these shift registers is the stored bit map in raster scan format.

As described, the hardware is capable of outputing the original matrix in any of its four mirror images. To obtain 90 degree rotations of these, an identical set of select and output hardware could be constructed along the horizontal axis. An alternative is to connect horizontal select line to the vertical ones along the diagonal of the matrix. In this way, the select and output logic is shared. Multiplexors on the inputs of the output register select either the vertically or horizontally selected words. Figure 6.3.2 illustrates the sequence of signals on the three control lines required to obtain a given orientation.

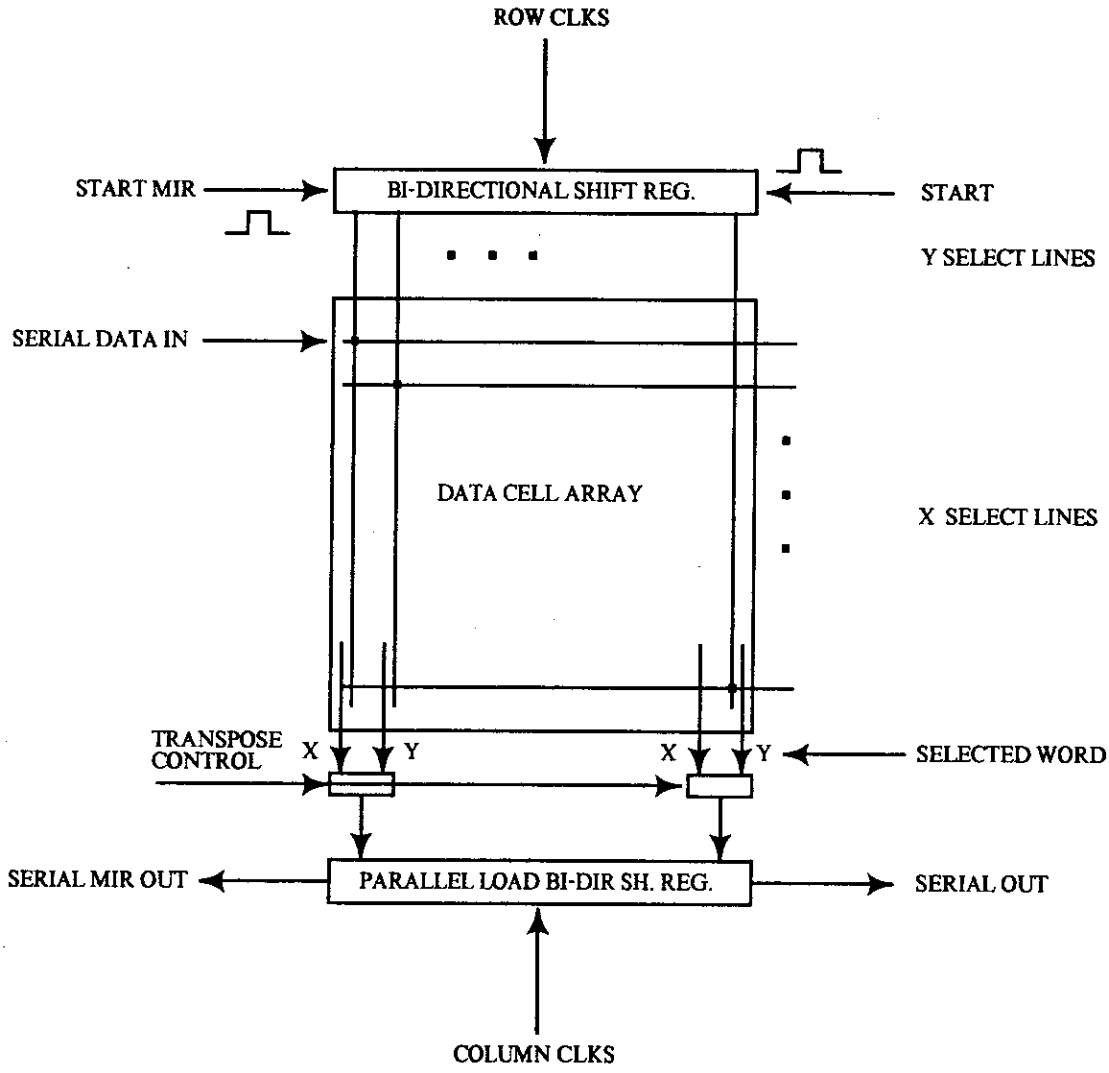

ROW CLKS
START MIR
BI-DIRECTIONAL SHIFT REG.
START
Y SELECT LINES
SERIAL DATA IN
DATA CELL ARRAY
X SELECT LINES
TRANSPOSE CONTROL
X
Y
X
Y
SELECTED WORD
SERIAL MIR OUT
PARALLEL LOAD BI-DIR SH. REG.
SERIAL OUT
COLUMN CLKS

Figure 6.3.1

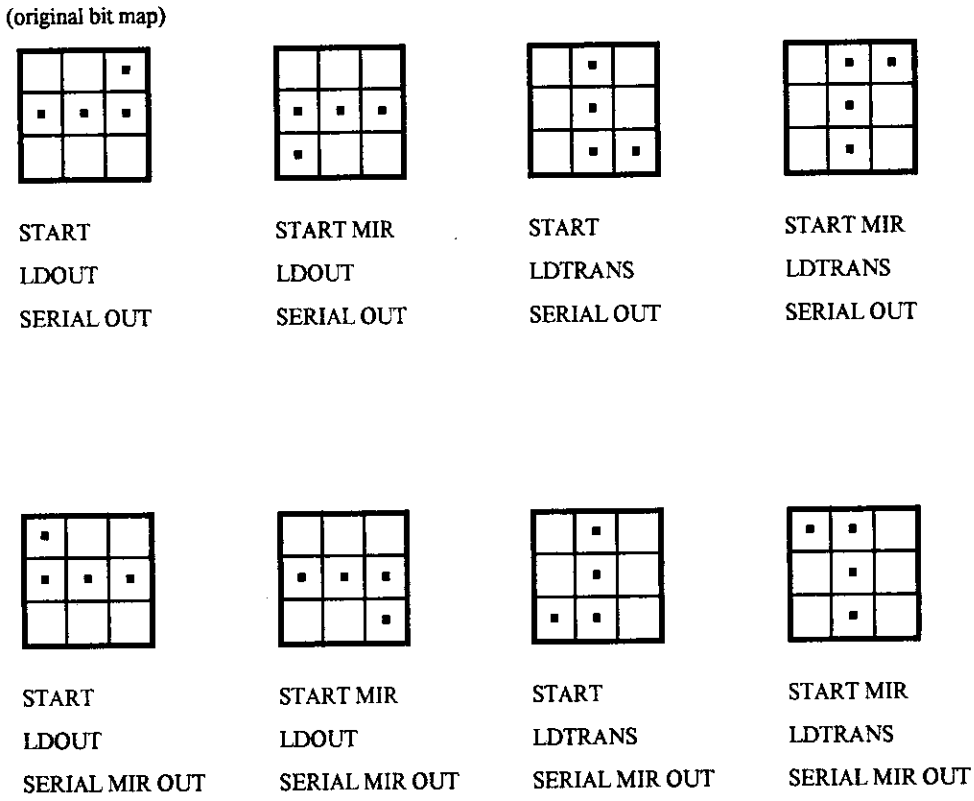
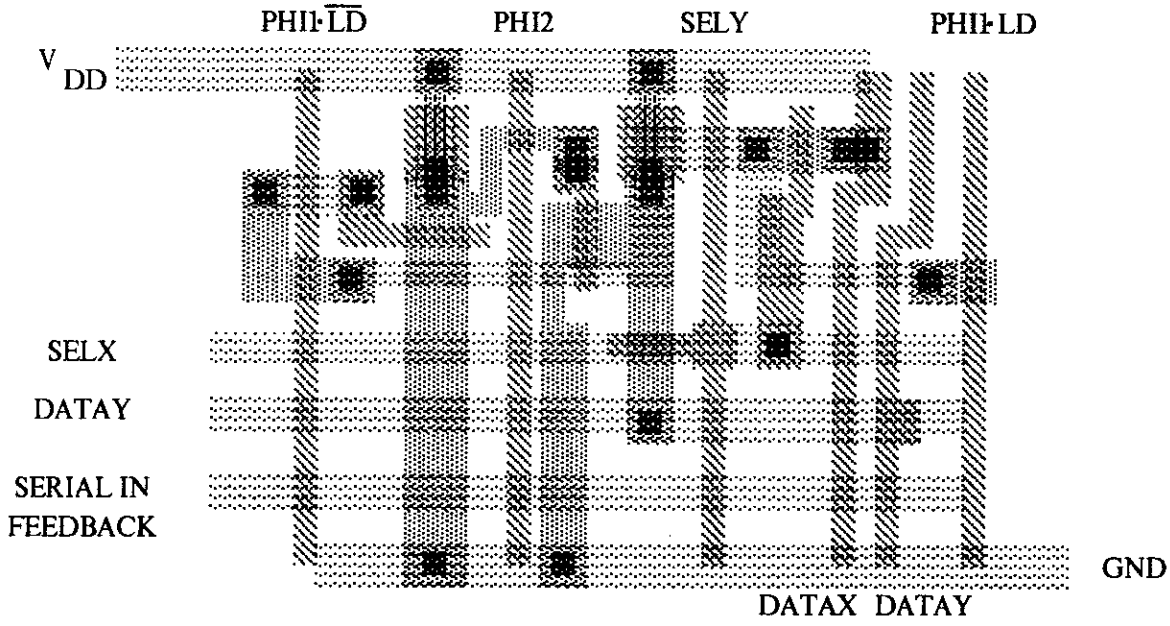


Figure 6.3.2 Available Transformations for a 3x3 bit matrix
Shown below each map are the control signals that are
asserted to obtain the indicated orientation.

6.3.5 Data Cell Design

Figure 6.3.3 illustrates the circuit used to realize the data cell. To minimize pinout, the entire array is loaded as a serial shift register by clocking $\varphi 1^*$LD and $\varphi 2$ while inputting the data on the DATA IN pin. The cell itself is simply a dynamic recirculating shift register. Clocks $\varphi 1^*\sim$LD and $\varphi 2$ run continously to refresh the cell. The select lines (X or Y) connect the output of the memory cell to an output bus orthogonal to the select line through a pass transistor. In order to have the DATAY buses come out at the bottom of the array, there are vertical DATAY lines also. Connections along the diagonal of the array map these horizontal buses to the vertical ones. In much the same way, SELECTX wires are connected along the diagonal to SELECTY wires. This means that a horizontal and vertical slice of the array are simultaineously selected. Both selected slices appear at the bottom of the array as eight pairs of DATAX, DATAY wires.



The circuit diagram corresponds to the topology of the layout. The array is loaded in a serpentine fashion, via a serial feedback wire that connects the last cell in a row to the first cell in the next row. This is also the same "raster scan format" of the original image. The serial output of the last data cell is connected to a test pad so that the loading of the data array can be tested independently of other functions.

The overall dimensions of a single cell are 45 by 70 square lambda. The cell is layed out such that V_{DD} and ground buses may be shared by mirror imaging alternate rows of the data cell array in order to conserve silicon area.

Since exactly one inverter in each cell is on at a time, the static power dissipation is:

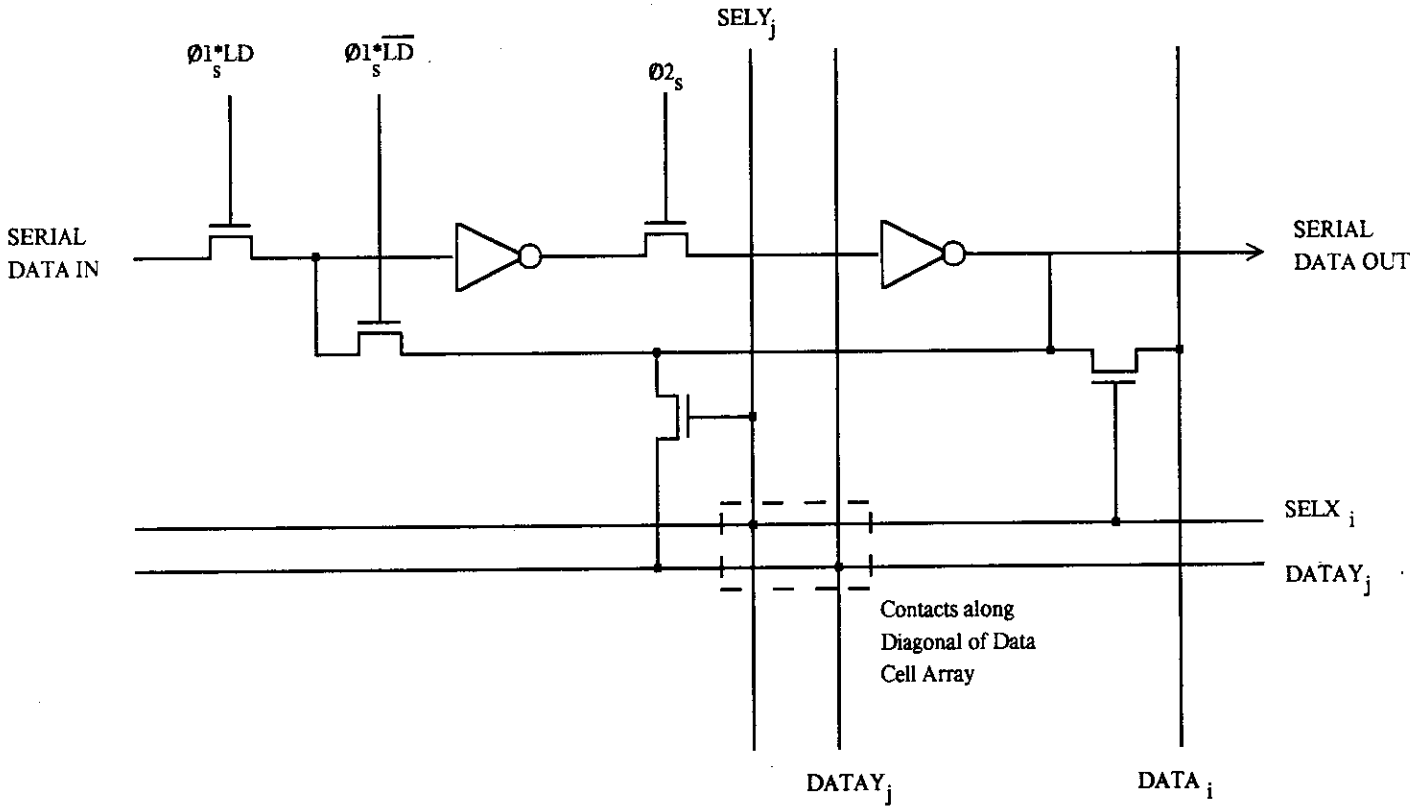


Figure 6.3.3 Data Cell Schematic

$$R = (1/3 + 3)\square * 10K\Omega/\square = 33K\Omega$$
$$I = V/R = 5v / 33K\Omega = .15\ mA$$

Thus, the entire array should consume about 10 mA.

A V_{DD} or Gnd bus wire must be capable of driving 16 cells, or 2.4 mA since it is shared by two rows. Assuming a maximum current density of 1 mA/μm^2, and 1 μm thick metal lines, the power buses should be capable of carrying:

$$4\lambda * 3\mu m/\lambda * 1\mu m * 1ma/\mu m^2 = 12\ mA$$

This indicates that the internal power buses are adequate. The internal power buses are connected along the vertical edges of the array with 10 λ wide wires which are in turn connected to power and ground pads at opposite corners of the array.

For a raster with 128 bits per line, the output shift register must be capable of shifting pixels out at a rate of 2 MHz. Row or column accesses of the array occur only one eighth as fast (for TV line rate). This means there is 4 μsec between row/column accesses.

For the data cell to be fast enough the output inverter pull-up transistors must source enough current to drive the relatively large capacitance of the DATAX and DATAY lines. The data cells on the diagonal of the array drive the most capacitance since they will be selected by both SELX and SELY lines and must drive both the DATAX and DATAY lines. Note that the DATAY lines consist of both a horizontal metal line and a vertical poly line. The total capacitance of these wires is:

$$C_l = 8[(3*70+27)*2.7*10^{-4}pf/\lambda^2 + ((2*48+8)+(2*35+3*12))*3.6*10^{-4}pf/\lambda^2]$$

$$= 1.1\ pf$$

The worst case for driving this capacitance is a low to high transistion. Refering to Chapter 2, page 17 of [Mead & Conway 1980] for the delay involved, we have

$$t = k\tau C_l/C_g$$

$$= 8 * 1ns * 1.1pf * 12*3.6*10^{-4}pf/\lambda^2$$

$$= 200ns$$

This is more than adequate for 128 pixels/line.

6.3.6 Select Register Design

The schematic of the select register is shown in figure 6.3.4. It consists of two shift registers which shift in opposite directions. The outputs at each stage are ORed together so that a pulse

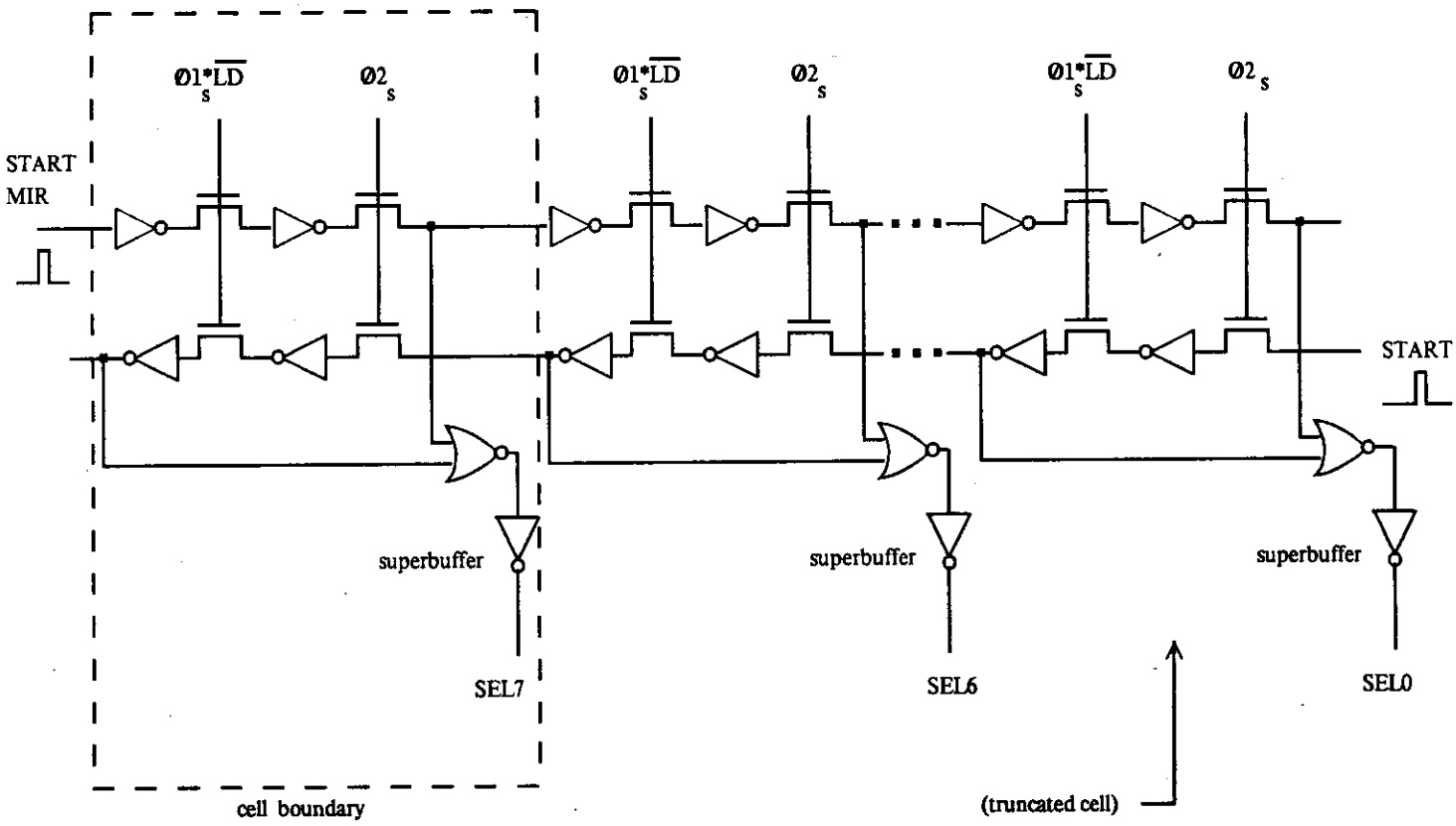


Figure 6.3.4 Select Register Schematic

propagating in either direction turns on the corresponding select line. The select lines are driven by super-buffers for speed. The output of both the top and bottom shift registers as well as SELY of each cell have large metal areas for wafer probing.

To initiate a read sequence the START or START MIR input is held high for one $\varphi 1$*LD, $\varphi 2$, clock cycle. This pulse then propagates through the select register . The same clocks that refresh the data cells are used to drive the select registers, since there is no harm in running them without a propagating start pulse. Figure 6.3.5 shows the layout of the basic select register cell. Bus lines to connect the various signals to the cells are included, to simplify the interconnect to the pads.

Each select register must drive a vertical poly and horizontal metal select line, plus their associated 16 pass transistors. The total capacitance of a select wire is:

$$C_l = 8[\ (45*2+16+16*4+14)*3.6*10^{-4}pf/\lambda^2 + 2*8*3.6*10^{-4}pf/\lambda^2 + (3*70+10)*2.7*10^{-4}pf/\lambda^2\ ]$$
$$= 1.46pf$$

This is driven by a super-buffer with the same size transistors as in the data cells, so that the time constant will be given by

$$RC = \tau C_l/C_g$$
$$= 1ns * 1.46pf / .043pf$$
$$= 33ns$$

since the rising and falling edges of the super-buffer's output will be symmetrical.

6.3.7 Output Register Design

The output register schematic is shown in figure 6.3.6. It consists of two parallel loading shift registers that run in opposite directions. A multiplexor at the top of the cell controls whether the original or 90 degree rotated image is output. Two outputs, one on the right, and one on the left of the output register are generated. Selecting which of these is used to drive the display provides mirroring control over the row/column of the array that is selected by the select register.

The clock lines that drive the output register are distinct from those that run the select register, since they must shift out an eight bit row/column for every row/column selected. These clocks are designated with an f subscript to denote "fast". Refering to the layout of this cell in figure 6.3.7, the gate of the pulldown transistor in top leftmost inverter, and the bottom right most transistor are omitted. This means that a logic 0 will be shifted out of the output register when no data is in it. This allows the output clocks to run continously to simplify the clock circuitry that drives them.

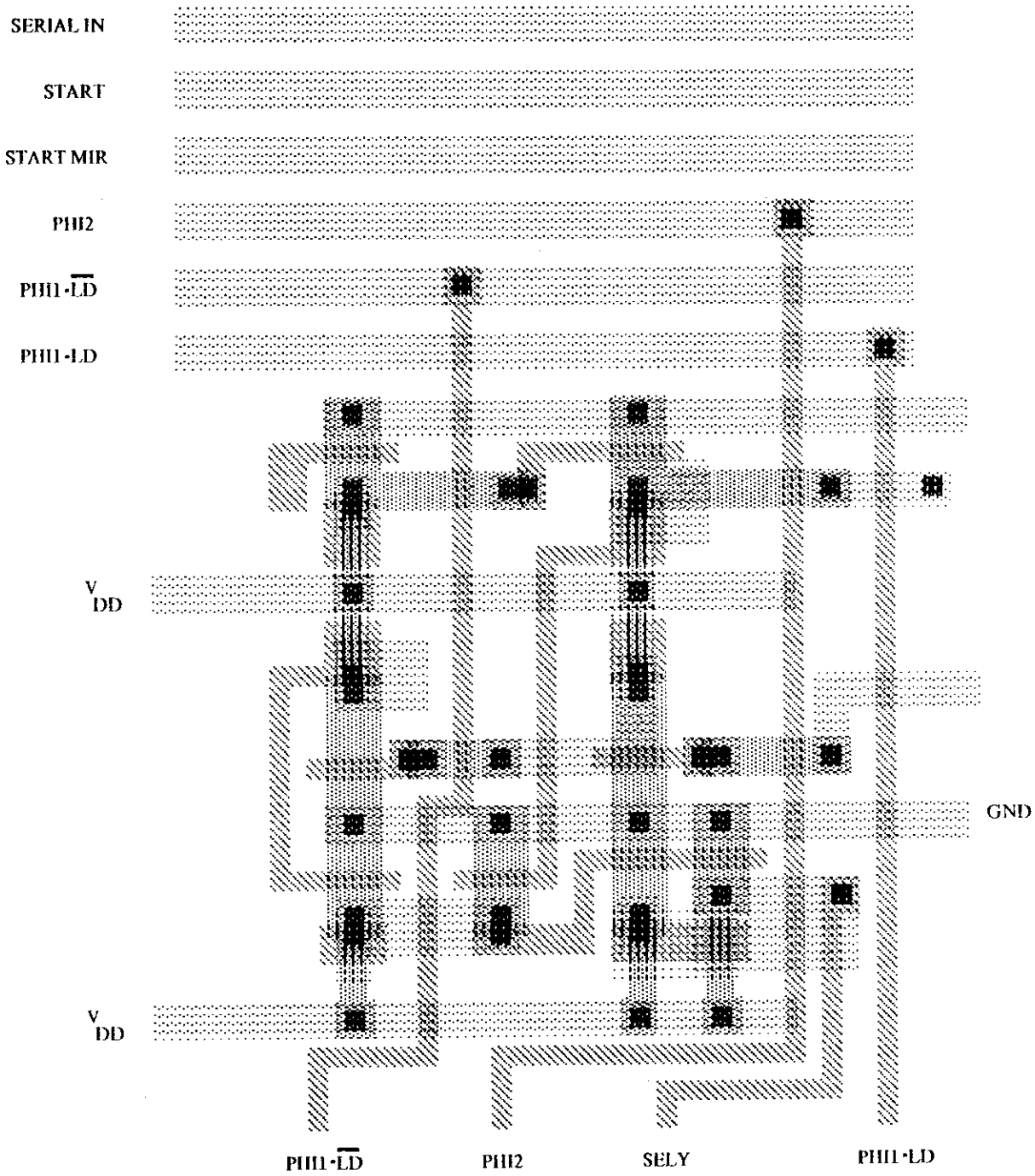


Figure 6.3.5 Select Register Layout

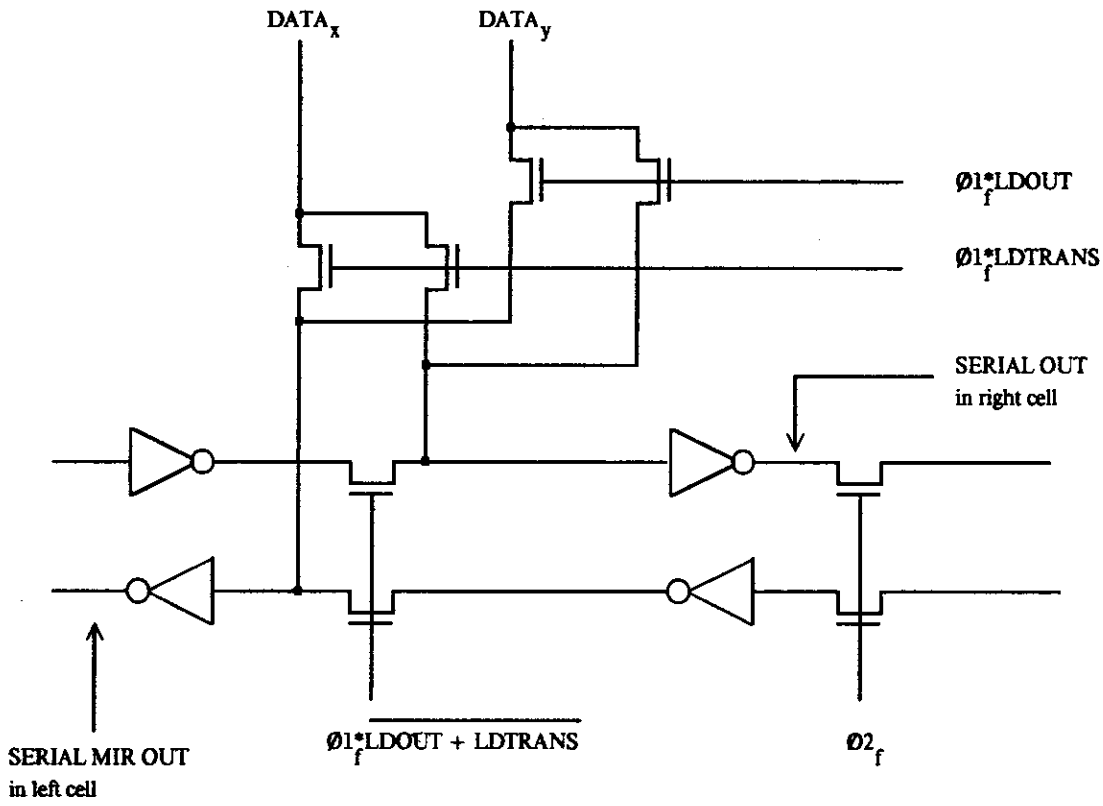


Figure 6.3.6 Output Register Cell Schematic

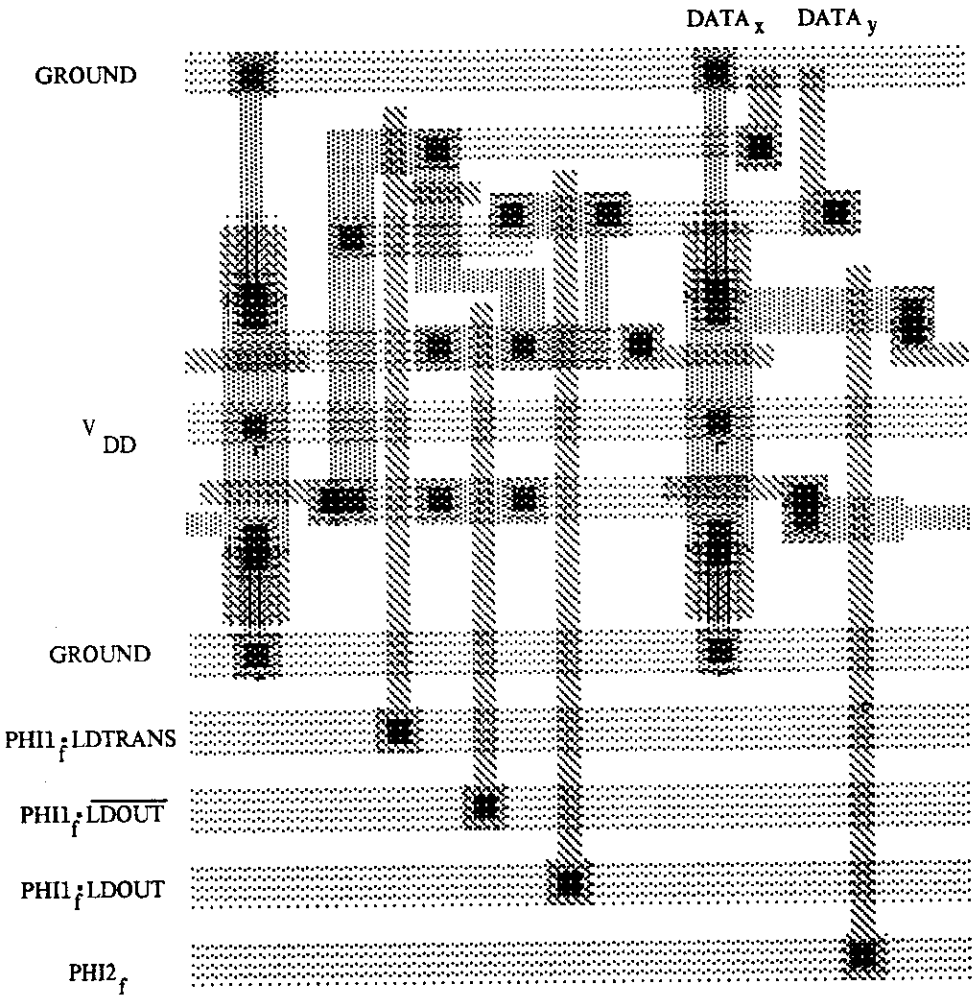


Figure 6.3.7 Output Register Cell Layout

6.3.8 Overall Timing

Figure 6.3.8 illustrates the timing of the various clocks. If the module is used in the graphics context described above, the slow clocks may be a two phase version of the horizontal sync pulse found in a raster type display. This occurs every 62.5 microseconds (for a 2MHz output bit rate), which is fast enough to keep the array cells refreshed. When external comparator logic determines that the first line of the object is being scanned, START or START MIR is held high. Similarly, a counter running at the pixel rate of the horizontal scan is compared to the X coordinate of the object, routing a $\varphi 1^*$~LD pulse to either $\varphi 1_f^*$LD or $\varphi 1_f^*$LDTRANS. Note that only equality must be detected (as opposed to "in region"), the clocks run continously, and do not require circuitry that counts out exactly eight clock cycles.

6.3.9 Testing

To test the fabricated chip, twelve TTL packages were assembled on a protoboard to generate all the necessary timing signals. Two 3-bit resistor ladder D/A converters were used to generate an 8x8-bit raster on an oscilloscope for displaying the image generated by the project. The Z-axis input of the oscilloscope was used to modulate the intensity of the displayed points.

Of the three bonded chips tested, two were 100% functional. Power consumption was 12 ma, as predicted. The project was also tested with the output shift register running at 2 MHz to test compatability with a 2 MHz CRT raster display system. At this speed the oscilloscope raster display was not functional so that the logic level of the output shift register was examined to determine if the transformations were correct. The two functional chips were successfully operated at this speed.

6.3.10 Conclusion

As is normally the case, this project was constraint by the limited time available and limits on the silicon area reserved for this chip. An obvious feature to add to this chip is the coordinate detection logic. This would not be to hard to implement, and would reduce the number of pins required.

References

[Dennard 1974]
R. H. Dennard et al, "Design of Ion-Implanted MOSFETS with Very Small Physical Dimensions," *IEEE J. Solid-State Circuits*, SC-9, October 1974.

[Grove 1967]
A. S. Grove, *Physics and Technology of Semiconductor Devices*, John Wiley and Sons, 1967.

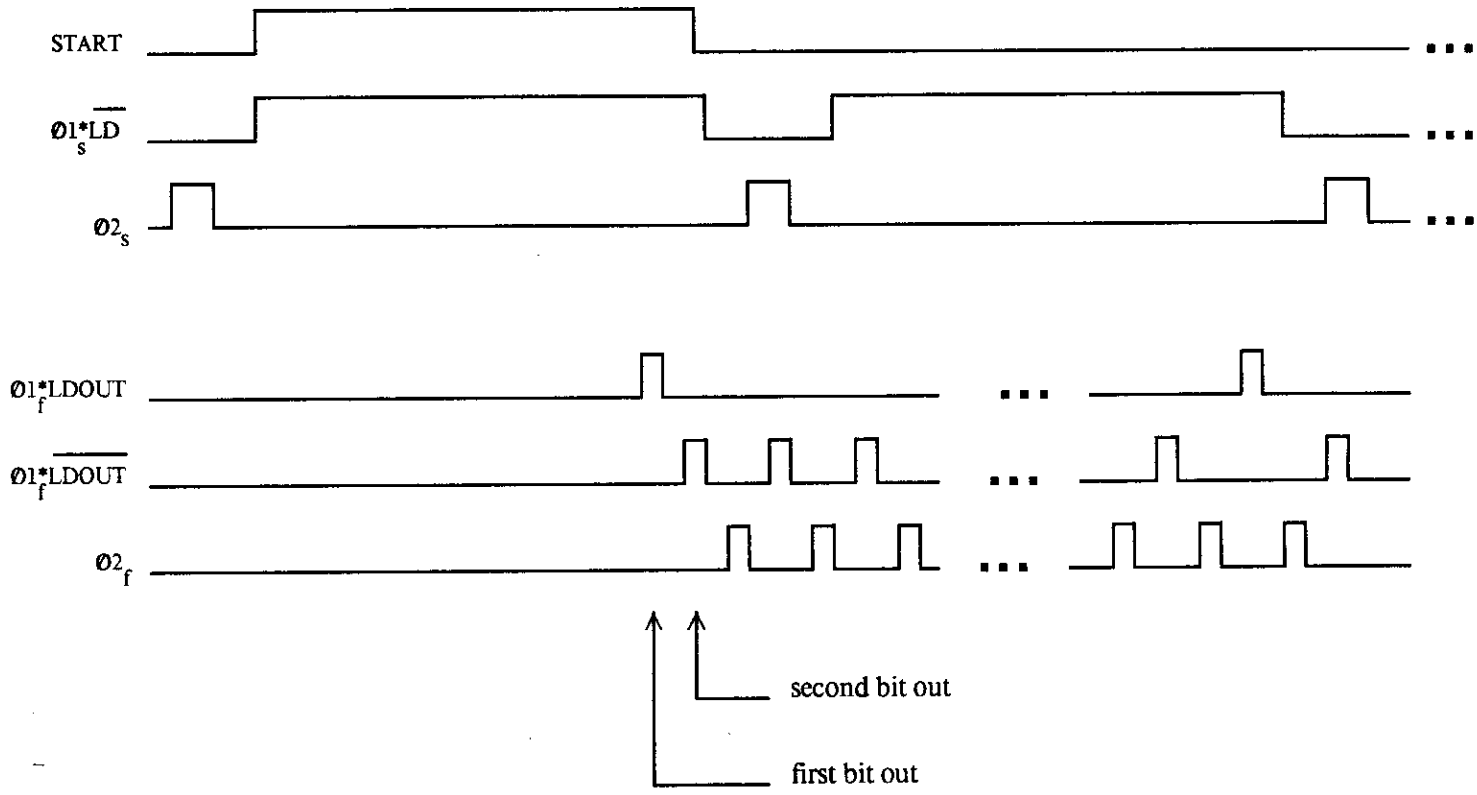


Figure 6.3.8 Timing Diagram for Readout Sequence

[Mead & Conway 1980]
C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA., 1980.

[Wang 1978]
P. P. Wang, "Device Characteristics of Short-Channel and Narrow-Width MOSFET'S, *IEEE Trans. on Electron Devices*, ED-25, July 1978, pp.779-786.

For additional example projects see:

[Clark 1980]
J. H. Clark, "A VLSI Geometry Engine for Computer Graphics", presented at the MIT conference on Advanced Research in Integrated Circuits, January 1980.

[Foster & Kung 1979]
M. Foster and H. T. Kung, "Design of Special-Purpose VLSI Chips: Example and Opinions", Carnegie-Mellon C.S. Tech. Report, September 1979.

CHAPTER 7

A CIF PRIMER

Caltech Intermediate Form (CIF) is a low-level graphics language for specifying the geometry of integrated circuits. Its purpose is to serve as a standard machine-readable definition of chip geometry that is an essential input to a chip fabrication facility. CIF is intended to serve as an unambiguous definition of geometries for designs ranging from a small student project up to the extremely complex VLSI chips anticipated in five years' time.

The most important feature of CIF is that it strives to be a *standard.* It defines geometry in a way that is not tied to specific pattern-generation equipment, to specific computer-aided design or artwork systems, or to specific fabrication technologies. Thus, a designer using *any* design technique may communicate, via a CIF file, with a fabrication facility using *any* pattern-generation equipment, an arrangement illustrated in Figure 7.1. This figure emphasizes that CIF is not a design data base, but rather an interchange format to couple the designer to the fabricator. Even though design systems and pattern-generation techniques may change, CIF can remain the standard way to specify a chip's geometry.

In addition to the benefits of CIF as a standard, it offers the following advantages:

1. It is easily generated and processed.
2. It has a clearly defined, unambiguous syntax.
3. It is compact due to its hierarchical symbol structure.
4. It is a text format, and is therefore machine-independent and transportable.
5. It is easily read by people as well as computers.
6. It provides a common denominator for designs, encouraging the sharing of cell libraries, etc.
7. It is independent of the process used to fabricate the wafer.

CIF 2.0 is already in use by a number of universities and industrial laboratories. Designs specified in CIF have been transported electronically between Xerox-PARC, Caltech, Carnegie-Mellon, Stanford, MIT, University of California at Berkeley, and the University of Washington using the ARPANET and other data links. As more universities begin to offer courses on VLSI design, we anticipate the use of CIF will grow substantially.

This chapter begins with a self-contained definition of CIF, including both detailed syntax and semantics of CIF files and some less formal conventions. The second part of the chapter contains some examples of how CIF can be generated and interpreted.

Portions of this chapter were written by R. F. Sproull and R. F. Lyon; sections 7.1 and 7.3 contain some material that is virtually identical to that found on pages 115-127 (section 4.5) of *Introduction to VLSI Systems* [Mead & Conway 1980], copyright 1980 by Addison-Wesley Publishing Company, Inc. and are reprinted by permission.

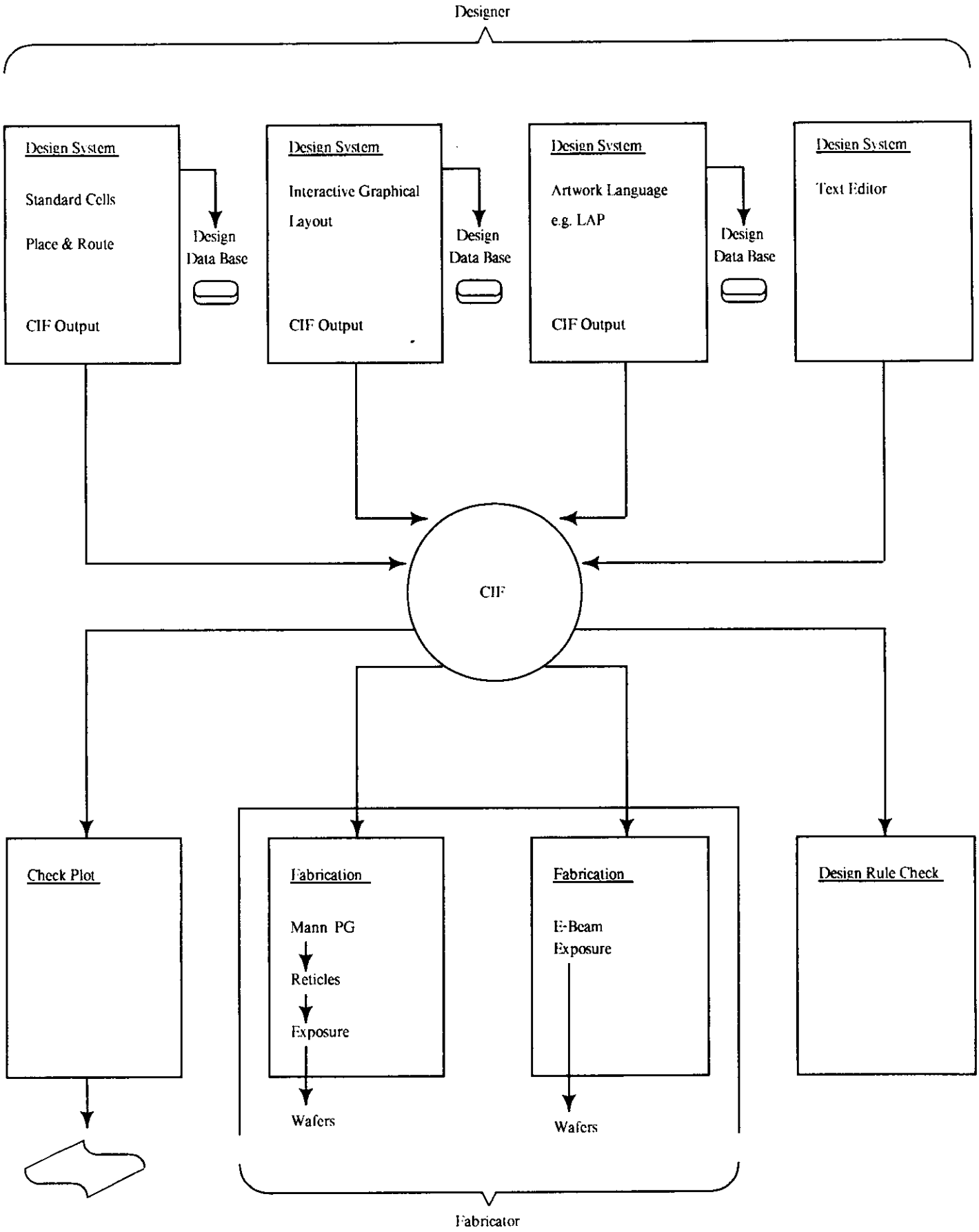

Designer
Design System
Standard Cells
Place & Route
CIF Output
Design Data Base
Design System
Interactive Graphical Layout
CIF Output
Design Data Base
Design System
Artwork Language e.g. LAP
CIF Output
Design Data Base
Design System
Text Editor
CIF
Check Plot
Fabrication
Mann PG
Reticles
Exposure
Wafers
Fabrication
E-Beam Exposure
Wafers
Design Rule Check
Fabricator

Figure 7.1

7.1 Definition of CIF 2.0

This section presents a concise definition of the CIF file format. Generally, the file may specify arbitrary geometries in an absolute coordinate system with a precision of 1/100th micron. These geometric patterns are expressed as a collection of *primitives*, such as boxes, wires, round flashes, and polygons of arbitrary shape. In addition, groups of these primitive items may be combined to form a *symbol*, which may be called to place *instances* of the symbol anywhere on the chip. The symbol may be mirrored, rotated, and translated at the time of the call. Although the symbol facilities add some complexity to CIF, they allow the geometry specification of a complex but regular design to be extremely compact.

The definition presented here is almost identical to that given in [Mead & Conway 1980]. Sections that are different or new are marked with an asterisk. It is intended that this chapter be the standard reference definition of CIF. As changes or clarifications become necessary, this section will be modified.

Portions of this section appear in [Mead & Conway 1980], and are reprinted by permission.

7.1.1 Syntax

A CIF file is composed of a sequence of characters in a limited character set. The file contains a list of commands, followed by an end marker; the commands are separated with semicolons. Commands are:

Command	Form
Polygon with a path	P path
Box with length, width, center, and direction (direction defaults to (1,0) if omitted)	B integer integer point point
Round flash with diameter and center	R integer point
Wire with width and path	W integer path
Layer specification	L shortname
Start symbol definition with index, a, b (a and b both default to 1 if omitted)	DS integer integer integer
Finish symbol definition	DF
Delete symbol definitions	DD integer
Call symbol	C integer transformation
User extension	digit userText
Comments with arbitrary text	(commentText)
End marker	E

A more formal definition of the syntax is given below. The standard notation proposed by Niklaus Wirth [Wirth 1977] is used: production rules use equals = to relate identifiers to expressions, vertical bar | for or, and double quotes " " around terminal characters; curly brackets { } indicate repetition any number of times including zero; square brackets [ ] indicate optional

factors (i. e., zero or one repetition); parentheses ( ) are used for grouping; rules are terminated by period. Note that the syntax allows blanks before and after commands, and blanks or other kinds of separators (almost any character) before integers, etc. The syntax reflects the fact that symbol definitions may not nest.

```
cifFile                = { { blank } [ command ] semi } endCommand { blank }.
command                = primCommand | defDeleteCommand |
                         defStartCommand semi { { blank } [ primCommand ] semi } defFinishCommand.
primCommand            = polygonCommand | boxCommand | roundFlashCommand | wireCommand |
                         layerCommand | callCommand | userExtensionCommand | commentCommand.

polygonCommand         = "P" path.
boxCommand             = "B" integer sep integer sep point [ sep point ].
roundFlashCommand      = "R" integer sep point.
wireCommand            = "W" integer sep path.
layerCommand           = "L" { blank } shortname.
defStartCommand        = "D" { blank } "S" integer [ sep integer sep integer ].
defFinishCommand       = "D" { blank } "F".
defDeleteCommand       = "D" { blank } "D" integer.
callCommand            = "C" integer transformation.
userExtensionCommand   = digit userText.
commentCommand         = "(" commentText ")".
endCommand             = "E".

transformation         = { { blank } ( "T" point | "M" { blank } "X" | "M" { blank } "Y" | "R" point ) }.

path                   = point { sep point }.
point                  = sInteger sep sInteger.

sInteger               = { sep } [ "-" ] integerD.
integer                = { sep } integerD.
integerD               = digit { digit }.

shortname              = c [ c ] [ c ] [ c ].
c                      = digit | upperChar.
userText               = { userChar }.
commentText            = { commentChar } | commentText "(" commentText ")" commentText.

semi                   = { blank } ";" { blank }.
sep                    = upperChar | blank.
digit                  = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
upperChar              = "A" | "B" | "C" | ... | "Z".
blank                  = any ASCII character except digit, upperChar, "-", "(", ")", or ";".
userChar               = any ASCII character except ";".
commentChar            = any ASCII character except "(" or ")".
```

7.1.2 Semantics

The fundamental idea of the intermediate form is to describe unambiguously the geometry of patterns for LSI circuits and systems. Consequently, it is important that all readers and writers of files in this form have exactly the same understanding of how the file is to be interpreted. This section is intended to contain sufficient information to allow unambiguous interpretation.

Measurements. The intermediate form uses a right-handed coordinate system shown in Figure 7.1.1, with x increasing to the right and y increasing upward. Directions and distances are always interpreted in terms of the front surface of the finished chip. The units of distance measurement are hundredths of a micron (μm).

Numbers. CIF uses numbers for several purposes: signed integers are used in coordinate measurements; unsigned numbers are used to specify widths, lengths, and symbol numbers. Signed numbers are restricted to lie in the range $-2^{24}+1 \leq x \leq 2^{24}-1$. Positive integers lie in the range $0 \leq x \leq 2^{24}-1$. These restrictions guarantee that a program reading CIF can represent any number in the file using a 25-bit one's or two's complement integer. *[Note: The restriction on the size of a number is not described in the CIF 2.0 description of [Mead & Conway 1980]. Some might argue that making such a restriction should require a change in CIF version number. However, since no programs yet written to generate or parse CIF will handle numbers larger than the size given here, adding the restriction induces no practical problems.]*

Directions. Rather than measure rotation by angles, CIF uses a pair of integers to specify a "direction vector." This technique eliminates the need for trigonometric functions in many applications, and avoids the problem of choosing units of angular measure. The first integer of a direction vector is the component of the direction vector along the x axis; the second integer is the component along the y axis. Thus a direction vector pointing to the right (the $+x$ axis) could be represented as direction (1 0), or equivalently as direction (17 0); in fact, the first number can be any positive integer as long as the second is zero. A direction vector pointing NorthEast (i.e., rotated 45 degrees counterclockwise from the x axis) would have direction (1 1), or equivalently (3 3), and so on.

7.1.2.1 Non-geometric Commands

User expansion: `5:NONSTANDARD DESIGN RULES: LAMBDA = 4.0;`

Several command formats (any command starting with a digit) are reserved for expansion by individual users; the authors of the intermediate form agree never to use these formats in future expansions of the standard format. For example, private expansions might provide for inserting instructions to a preprocessor that will be ignored by any program reading only standard intermediate form constructs; or recording ancillary information or data structures (e.g., circuit diagrams, design-rule check results) that are to be maintained in parallel with the geometry specified in the style of the intermediate form.

Users should be cautious about using local extensions that will render the CIF file meaningless to a program that cannot process the extension properly. For example, a common extension is to request that another file be "inserted" at this point in the processing, thus simplifying the use of symbol libraries. A standard CIF-reader will not insert the symbol definitions, causing the

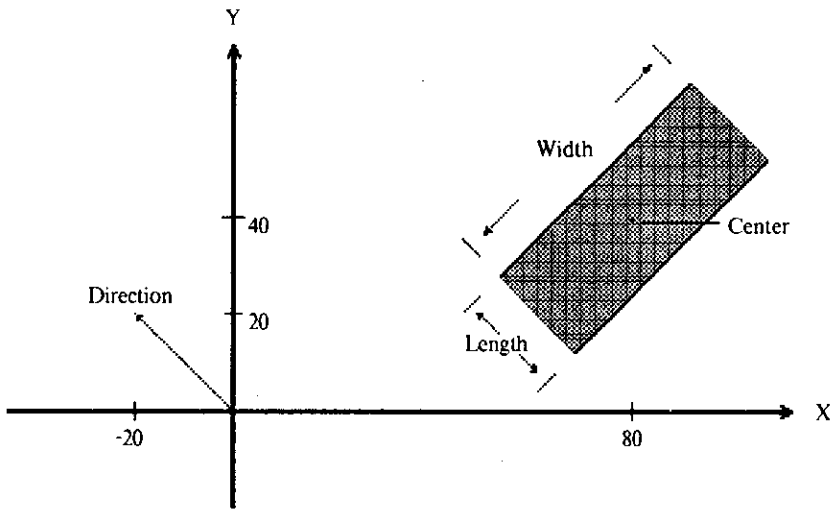


Figure 7.1.1 Box Representation in Intermediate Form

interpretation of the CIF file to be altered.

Comments: (HISTORY OF THIS DESIGN:);

The comment facility is provided simply to make the file easier to read. Note that a comment is a command: it must be terminated with a semi-colon, and can begin only in those places where commands can begin. The comment command can be used to deactivate any number of commands by simply enclosing them within a pair of parentheses, even if they already include balanced parentheses.

End Command: End of file.

The final E signals the end of the CIF file.

7.1.2.2 Geometric Primitives

The various primitives that specify geometric objects are not intended to be mutually exclusive or exhaustive. CIF may be extended occasionally to accommodate more exotic geometries. At the same time, it is not necessary to use a primitive just because it is provided. Notice in the examples below that lower case comments and other characters within a command are treated as blanks, and that blanks and upper case characters are acceptable separators.

Boxes: Box Length 25 Width 60 Center 80,40 Direction -20,20; (or B25 60 80 40 -20 20;);

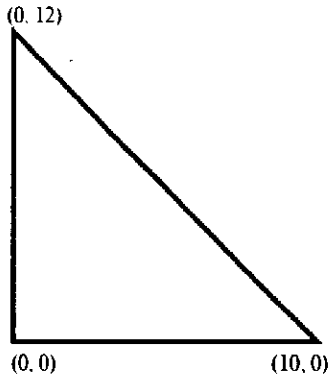
The fields that define a box are shown graphically in Figure 7.1.1. *Center* and *direction* specify the position and orientation of the box, respectively. *Length* is the dimension of the box parallel to the *direction*, and *width* is the dimension perpendicular to the *direction*. The *direction* may be omitted, in which case it will be defaulted to (1 0).

Polygons: Polygon A 0,0 B 10,20 C -30,40; (or P0 0 10 20 -30 40;);

A polygon is an enclosed region determined by the vertices given in the path, in order. For a polygon with n sides, n vertices are specified in the path -- the edge connecting the last vertex with the first is implied; see Figure 7.1.2a. Polygons with "holes" in them may be specified by the artifice illustrated in Figure 7.1.2b: the hole is part of the polygon boundary, and is joined to the outside boundary with a "channel" of zero width.

The polygon boundary may be self-intersecting, i.e., an edge may cross another edge. In this case, a point is on the interior of the polygon if the "winding number" (sometimes called "wrap number") of its boundary is non-zero. The wrap number can be visualized as follows: for each edge in the polygon, calculate the angle in radians subtended by the edge as viewed from the point; this angle will be positive if the edge is directed counter-clockwise with respect to the point, and negative if directed clockwise. Sum the angles for all edges; the wrap number is $sum/(2\pi)$. The wrap-number calculation is illustrated in Figure 7.1.3a. Figure 7.1.3b shows a polygon and the wrap

(a)



Polygon: P 0.0 10.0 0.12

(b)

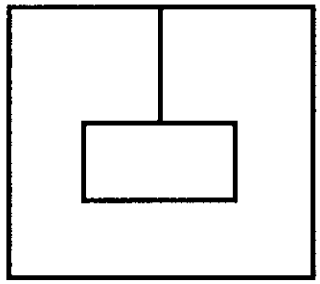


Figure 7.1.2

numbers of corresponding areas. Interior points (boundary wrap number non-zero) are shaded. *[This paragraph clarifies the definition of polygons in [Mead & Conway 1980].]*

Flashes: RoundFlash Diam 200 Center -500,800; (or R200 -500 800;);

This primitive calls for a circular shape, specified by its diameter and the location of its center (Figure 7.1.4a).

Wires: Wire Width 50 A 0,0 B 10,20 C -30,40; (or W50 0 0 10 20 -30 40;);

It is sometimes convenient to describe a long, uniform width run by the path along its centerline. We call this construct a wire (see Figure 7.1.4b). An ideal wire is the locus of points within one half-width of the given path. Each segment of the ideal wire therefore includes semicircular caps on both ends. Connecting segments of the wire is a transparent operation, as is connecting new wires to an existing one: the semicircular overlap ensures a smooth connection between segments in a wire and between touching wires.

Layer specification: Layer ND nmos diffusion; (or LND;);

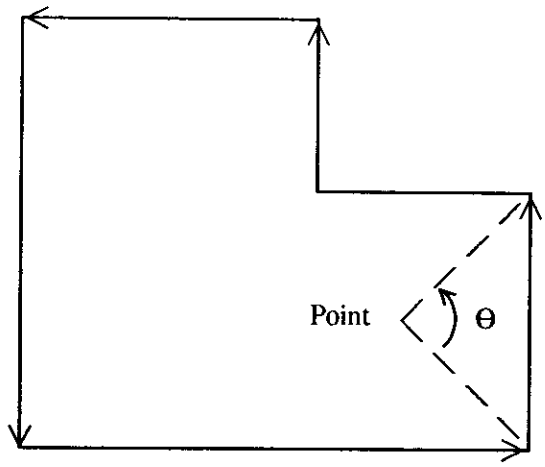
Each primitive geometry element (polygon, box, flash, or wire) must be labeled with the exact name of a fabrication mask on which it belongs. Rather than cite the name of the layer for each primitive separately, the layer is specified as a "mode" that applies to all subsequent primitives, until the layer is set again. It is illegal to specify a geometric primitive without having previously set the layer mode. Layer mode is preserved across symbol definitions and calls, which are discussed later.

The argument to the layer specification is a short name of the layer. Names are used to improve the legibility of the file and to avoid interfering with the various biases of designers and fabricators about numbers (one person's "first layer" is another's "last"). It is important that layer names be unique, so that combining several files in intermediate form will not generate conflicts. The general idea is that the first one or two characters of the name denote the technology, and the remainder is mnemonic for the layer. At present, the following layers are defined:

ND	NMOS Diffusion
NP	NMOS Polysilicon
NC	NMOS Contact cut
NM	NMOS Metal
NI	NMOS depletion mode Implant
NB	NMOS Buried contact
NG	NMOS overGlass openings

New layer names will be defined as needed in order to accommodate CMOS-bulk, CMOS-SOS and other technologies. Layer names will be added to the CIF definition without introducing incompatibilities in the format itself.

(a)



(b)

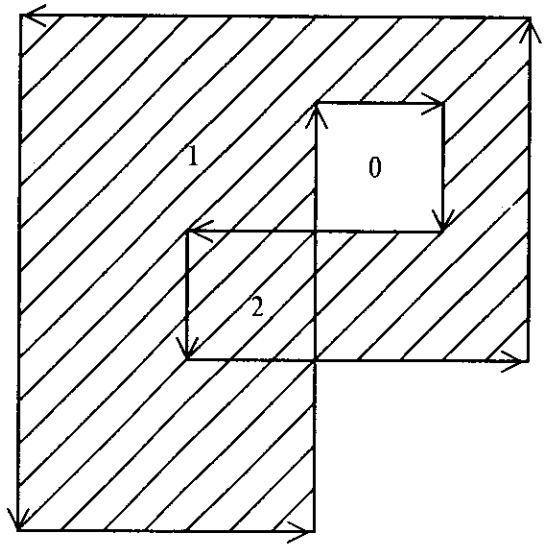
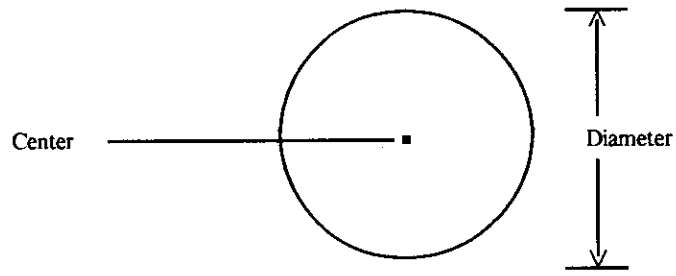


Figure 7.1.3

(a)



(b)

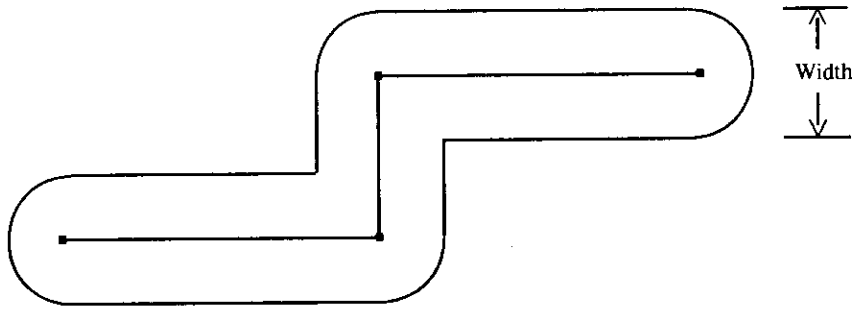


Figure 7.1.4

7.1.2.3 Symbols

Because many LSI layouts include items that are often repeated, it is helpful to define often-used items as "symbols." This facility, together with the ability to "call" for an instance of the symbol to be generated at a specific position, greatly reduces the bulk of the intermediate form.

The symbol facilities are deliberately limited, in order to avoid mushrooming difficulties of implementing programs that process CIF files. For example, symbols have no parameters; calling a symbol does not allow the symbol geometry to be scaled up or down; there are no direct facilities for iteration. If the symbol mechanism is not adequate for some application, the desired geometry can still be achieved with less use of symbols, and more use of explicit geometrical primitives. CIF files need not use symbols at all, although a rather large file will probably result.

Defining symbols: Definition Start #57 A/B = 100/1; ... ; Definition Finish; (or DS57 100 1; ... ;DF;);

A symbol is defined by preceding the symbol geometry with the DS command, and following it with the DF command. The first argument of the DS command is an identifying *symbol number*, unrelated to the order of listing of symbol definitions in the file.

The mechanism for symbol definition includes a convenient way to scale distance measurements. The second and third arguments to the DS command are called a and b respectively. As the intermediate form is read, each distance (position or size) measurement cited in the various commands (polygons, boxes, flashes, wires and calls) in the symbol definition is scaled to (a*distance)/b. For example, if the designer uses a grid of 1 micron, the symbol definition might cite all distances in microns, and specify a = 100, b = 1. Or the designer might choose lambda (characteristic fabrication dimension) as a convenient unit. This mechanism reduces the number of characters in the file by shrinking the integers that specify dimensions and may improve the legibility of the file. It provides neither scaling nor the ability to change the size of a symbol called within the definition.

Definitions may not nest. That is, after a DS command is specified, the terminating DF must come before the next DS. The definition may, however, contain calls to other symbols, which may in turn call other symbols. Additional information about symbol definitions is provided in *Symbol Interpretation Rules*, below.

Calling symbols: Call Symbol #57 Mirrored in X Rotated to -1,1 then Translated to 10,20;

The C command is used to call a specified symbol and to specify a transformation that should be applied to all the geometry contained in the symbol definition. The call command identifies the symbol to be called by its symbol number, established when the symbol was defined.

The transformation to be applied to the symbol is specified by a list of primitive transformations given in the call command. The primitive transformations are:

T point	Translate the current symbol origin to this point.
M X	Mirror in X, i.e., multiply X coordinate by -1.
M Y	Mirror in Y, i.e., multiply Y coordinate by -1.
R point	Rotate symbol's x axis to this direction.

Intuitively, each coordinate given in the symbol is transformed according to the first primitive tranformation in the call command, then according to the second, etc. Thus "C1 T500 0 MX" will first add 500 to each *x* coordinate from symbol 1, then multiply the *x* coordinate by -1. However, "C1 MX T500 0" will first multiply the *x* coordinate by -1, and then add 500 to it: the order of application of the transformations is therefore important. In order to implement the transformations, it is not necessary to perform each primitive operation separately; the several operations can be combined into one matrix multiplication (see [Newman & Sproull 1979]).

Symbol calls may nest; that is, a symbol definition may contain a call to another symbol. When calls nest, it is necessary to "concatenate" the effects of the transformations specified in the various calls (see [Newman & Sproull 1979]).

The layer mode is preserved across symbol calls and definitions. Thus, in the sequence:

```
LNM;
R6 20 0;

C 57 T45 13;

DS 114;
( ... definition of symbol 114);
DF;

LNM;
R3 0 0;
```

the second LNM is not necessary, regardless of the specification of symbols 57 and 114.

Deleting symbol definitions: Delete Definitions greater than or equal to 100; (or DD100;);

The DD command signals the program reading the file that all symbols with indices greater than or equal to the argument to DD can be "forgotten" – they will not be instantiated again. This feature is included so that several CIF files can be appended and processed as one. In such a case, it is essential to delete symbol definitions used in the first part of the file both because the definitions may conflict with definitions made later and because a great deal of storage can usually be saved by discarding the old definitions.

The argument to DD that allows some definitions to be kept and some deleted is intended to be used in conjunction with a standard "library" of definitions that a group may develop. For example, suppose we use symbol indices in the range 0 to 99 for standard symbols (pullup transistors, contacts, etc.) and want to design a chip that has 2 student projects on it. Each project

defines symbols with indices 100 or greater. The CIF file will look like:

```
(Definitions of library symbols);
DS 0 100 1;
( ...definition of symbol 0 in library);
DF;
DS 1 100 1;
( ...definition of symbol 1);
DF;
( ...remainder of library);

(Begin project 1);
DS100 100 1;
( ...first student's first symbol definition);
DF;
...
DS109 100 1;
( ...first student's main symbol definition);
DF;
C109 T403 -110; (call on first student's main symbol);

DD100; (Preserve only symbols 1 to 99);

(Begin project 2);
DS100 100 1;
( ...second student's first symbol definition);
DF;
...
DS113 100 1;
( ...second student's main symbol definition);
C1 T-3 45; (Call on library symbol, still available);
DF;
C113 T401 0; (call on second student's main symbol);

E
```

*7.1.2.4 Symbol Interpretation Rules**

The use of symbols in CIF requires a careful definition of the rules for interpreting a CIF file. We shall try to give a precise description without resorting to formalisms. The design of CIF and of the symbol facilities permits a straightforward interpretation scheme that processes commands in the file in order, in one pass over the file.

If no symbol facilities are used, the interpretation rule is very simple. Whenever a Layer command is parsed, the layer name it specifies is recorded as the *current layer.* Whenever a geometric primitive command is found, it can be "output" on the current layer. The kind of output generated will vary with the application of CIF: if we are making a check-plot, the primitive will be traced on a plotter, perhaps using a colored pen selected by the current layer.

Symbol facilities require that this simple interpretation rule be modified somewhat. The CIF file is still scanned once, from beginning to end. When the beginning of a symbol definition is encountered (DS command), subsequent commands are not "output" directly, but are instead retained, in the same order they appear in the file, as a *symbol definition.* When the end of the

symbol definition is encountered (DF command), the retained definition is entered in a *symbol table.* The symbol table maps a *symbol number* (the first argument following a DS command) into the corresponding *symbol definition*, a list of commands.

As the CIF file is scanned, we may also encounter commands that are not embedded in symbol definitions; these are *executable commands.* They are "output" directly, just as in the case mentioned at the outset where no symbol facilities are used. If the executable command is a Call on a symbol, the symbol definition is found by looking in the symbol table for the defintion with the corresponding *symbol number.* The interpreter now begins processing the commands saved as part of the symbol definition, with the additional need to apply to each geometrical primitive the transformation specified in the call. If, as the symbol is being interpreted, another Call command is encountered, its symbol definition is looked up in the symbol table, and the interpreter starts processing its commands, applying a transformation that is the concatenation of the transformations specified in the two calls. When all commands in the symbol definition have been processed, we resume interpretation of the first symbol where we left off, and so on. Eventually, the interpretation of the executable Call command is complete, and we continue scanning the file.

This model of the interpretation procedure provides intuitive answers to a number of common questions about the use of symbols:

1. Symbol definitions may occur in any order in the CIF file. The only restriction is that a symbol must be defined (i.e., entered into the symbol table) before its interpretation becomes necessary. Thus, the definitions for symbols 0 and 1 in the following file could be in either order:

```
DS 0 100 1;
( ...definition of symbol 0 );
C 1 T 14,32; ( call on symbol 1 );
DF;

DS 1 100 1;
( ...definition of symbol 1 );
DF;

C 0 ; ( Executable call; both symbols defined );
```

The call on symbol 1 inside symbol 0 is legal, because symbol 1 will be defined before symbol 0 is interpreted.

2. Deleting symbol definitions. The effect of the DD command is to scan the symbol table and remove any symbol definition with an identifying symbol number greater than or equal to the argument to DD. This may have the puzzling effect of retaining symbol definitions that *call* symbol numbers that, after the deletion is performed, do not exist. These are called "dangling references." It is helpful to issue a warning message if any dangling references remain after deleting symbols. The message should read: "Warning: dangling references after DD." *It is not considered normal practice to leave references dangling intentionally.*

3. Multiple symbol definitions. If a symbol definition is encountered that defines a symbol number already recorded in the symbol table, the old definition is replaced by the new one, and a warning message is issued. The message should read: "Warning: symbol *n* redefined." After the new symbol is entered, dangling references may remain, as described above; a warning message may be helpful. *Again, it is not considered normal practice to redefine symbols in CIF.*

7.1.3 The Relationship Between CIF and Fabricated Chips*

The relationship between geometric patterns specified in a CIF file and the patterns on a chip fabricated from the file cannot be controlled precisely. To attempt to do so would make the transformation from a CIF file to mask patterns difficult or even impossible.

CIF specifies the "nominal fabricated geometry" on the chip. By speaking of "fabricated geometry," we mean that the file specifies what the patterns should look like after fabrication. By "nominal" we mean that any given chip will depart in various ways from the exact CIF specification, due to variations occurring throughout the manufacturing process: alignment variations, slight variations in widths of objects, etc.

In most cases, extremely precise correspondence between the CIF geometry and the fabricated geometry may not be necessary. CIF will usually be used in conjunction with some *geometric design rules* associated with a particular IC technology. Adhering to these design rules precludes geometries that cannot tolerate small distortions during fabrication. A designer expects that a layout conforming to the rules will be fabricated sufficiently precisely to operate properly. Thus the design rules and the choice of a characteristic dimension are a measure of the fidelity with which the lithography and fabrication processes can manufacture the geometry specified in the CIF file. CIF that does not meet design rules cannot be guaranteed to be manufactured properly.

The following paragraphs expand on the notion of "nominal fabricated geometry," and try to answer the most common questions that arise about the interpretation of CIF geometry:

1. Features below the resolution of the manufacturing process. Although a CIF file may contain geometric shapes of very small size (.01 μm), the manufacturing process will not reproduce features smaller than a certain size.

When describing the relationship between CIF geometry and what appears on the chip, we often need to discuss the characteristic dimension of the lithographic and fabrication processes. In the design rules for NMOS given in [Mead & Conway 1980], this dimension is denoted by λ. In those rules, the single parameter is quite closely related to the geometric errors that are introduced in the manufacturing process. Design rules for other technologies may employ other parameters, which may not reflect only geometric fabrication limitations. In our discussion, we shall use λ to express the geometric fabrication tolerances, with the understanding that a similar parameter could be devised for technologies other than NMOS.

By following the geometric design rules for a particular process, the designer will be prevented from specifying patterns too small to fabricate. It should be pointed out that common right-angle corners technically constitute design-rule violations, because they require fabrication of an object with a width dimension less than λ (Figure 7.1.5a). Even if a design-rule checker is instructed to overlook such errors, the designer cannot expect perfect construction of the corner; a hypothetical result of manufacturing the shape with a process of minimum feature size 2λ is shown in Figure 7.1.5b.

2. Compensation for line-width distortion. The CIF file specifies the sizes that features should have once fabricated, which are not necessarily equivalent to the sizes of these features on a mask or to the sizes specified to pattern-generation equipment. Depending on the amount of etching done in certain process steps, on the differences between positive and negative resist steps and other effects, line widths extracted from the CIF file may need to be broadened or narrowed slightly before being passed to pattern-generation equipment. This compensation usually applies equally to all shapes, and thus requires a geometric "shrink" or "expand" to be applied to all geometric figures in the CIF file before generating a mask.

3. Connections. Two shapes on the same layer are assumed to be connected electrically if they abut one another and meet the geometric design rules. The shapes need not overlap in order to be connected. Figure 7.1.6a shows a legal connection made by abutting two shapes; the design in Figure 7.1.6b is illegal because a 2λ minimum-width design rule is violated at the joint.

4. Overlap. CIF allows objects on the same layer to overlap arbitrarily. Unfortunately, not all pattern-generation processes accommodate overlap. Repeated overlapping flashes on most present-day optical mask-generation equipment will overexpose the reticle, with a consequent "blooming" of the desired shape due to scattering and flare. A CIF file should be processed to remove overlaps before generating instructions for a pattern generator with this problem. Because this process depends on details of the pattern-generation process used, it should be the responsibility of the fabricator, not of the designer.

7.1.4 Common Conventions for Using CIF*

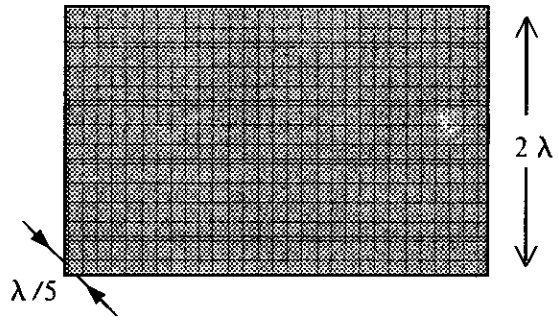
CIF was designed to be used in a variety of ways in the preparation of LSI artwork. The definition of legal CIF does not reflect all of the conventions that make CIF easy to use. This section outlines some conventions, and gives names to them. By giving names to the common ways to use CIF, we anticipate individual designers or design systems may simply adopt one or more of the conventions.

1. *Version comment convention.* This convention requires a CIF file to contain a comment near the beginning that describes the version number of the CIF format used in the file. For example:

```
(CIF 2.0);
```

This convention clearly labels the file with the assumptions used to generate it. It is also wise to

(a)



(b)

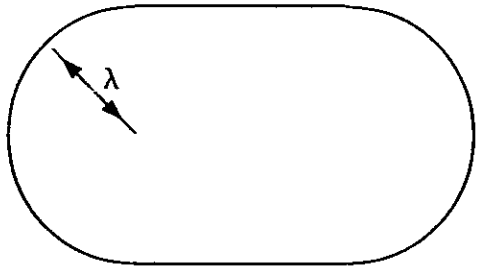
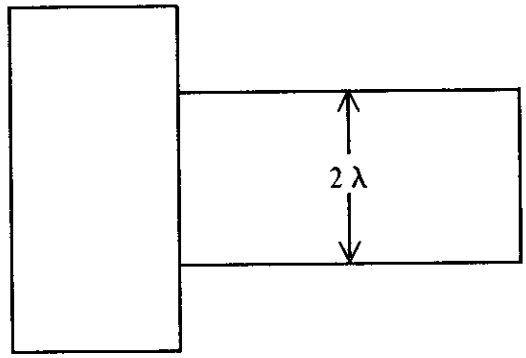


Figure 7.1.5

(a)



(b)

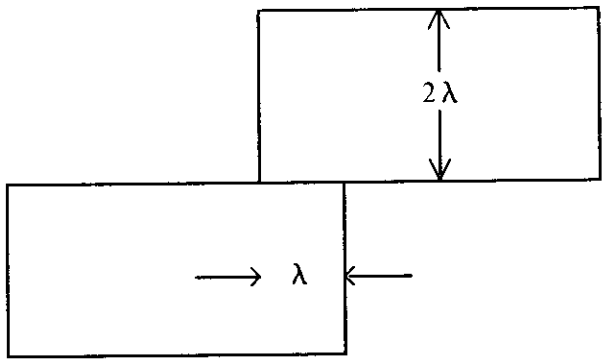


Figure 7.1.6

identify the designer and his or her organization at the beginning of the CIF file, perhaps including a date and design revision code.

2. *Fabrication comment convention.* A CIF file specifies only a portion of the information required to fabricate a chip. In addition to the geometry in the file, we need to specify some parameters of the fabrication process. This convention places such parameters in a comment near the beginning of a file; eventually a fully digital interface to fabrication may define these parameters precisely in machine-readable form. Until it becomes clear exactly what parameters must be supplied, the comment can contain arbitrary text:

```
(FAB nMOS silicon gate, Mead&Conway design rules
    lambda=3 microns; design can be scaled if necessary
    from lambda=2 min to lambda=5 max );
```

The comment explains that the geometry is for nMOS and has been defined using design rules in [Mead & Conway 1980] and that the parameter λ in those rules is 3 microns. It also says that the entire design could be scaled over a range from $\lambda=2$ to $\lambda=5$ microns in order to match the properties of a particular fabrication line. Indicating some flexibility in actual fabrication parameters reduces the constraints on combining separate experiments on a multi-project chip or on identifying a fabrication line that can process the design.

3. *No call convention.* Some CIF files will not use the symbol facilities of CIF at all; these are said to be "no call files." They contain no instances of the DS, DF, DD, or C commands.

4. *One call convention.* A common way to use CIF is to define a layout by an arbitrary number of symbol definitions, following by a *single* Call command. A CIF file conforming to this "one call convention" might look like:

a. Symbol definitions for "primitive" components, such as transistors, contacts, pads, etc.

b. Symbol definitions for building-blocks such as PLA cells, adder cells, register cells, pad drivers, super-buffers, etc.

c. Symbol definitions for functional parts of the design: a particular adder, PLA or shift register.

d. A single symbol definition that specifies the entire design by calling other symbols (presumably mostly those in category c), often using additional wiring to connect functional parts.

e. A single call on the symbol defined in (d). The transformation in the call determines where on the chip the entire design will lie.

The ordering of symbol definitions (a,b,c,d) is not critical; what is essential for this convention is that the only executable CIF command is the single call at the end (e).

This convention greatly simplifies the assembly of several projects on a single chip. By altering the transformation in the single call, the project can be placed in the correct position and orientation on the actual chip. Moreover, several projects can be combined into a single CIF file as follows:

1.1 Symbol definitions for project 1 (a,b,c,d).

1.2 The call for project 1, modified suitable to place the project properly.

1.3 A DD 0; command to cancel all symbol definitions associated with project 1.

2.1 Symbol definitions for project 2, similar to 1.1 above.

2.2 Call for project 2, similar to 1.2 above.

2.3 DD 0; to cancel project 2 symbols.

.

.

n.1, n.2, n.3 As many projects as you like.

s.1, s.2, s.3 A "project" to define scribe lines, test patterns, alignment marks, etc.

E The final "end" command.

It is possible to convert any CIF file into a "one call" file. For example, a no call file can be made into a single symbol by placing at the beginning a DS 0; command and DF; and C 0; commands before the End command. Files that already contain symbol definitions and executable commands interspersed require more complex modifications to make a "one call" file (e.g., renumbering symbols to create unique numbers, removing DD commands, etc.).

5. *x-Skeletal connectivity convention.* CIF files generated using this convention require that two objects on the same layer that are intended to make electrical contact must overlap by at least $2x$. Thus we speak of λ-skeletal connectivity convention or a 1-micron-skeletal connectivity convention. This requirement for connection is stronger than the abutting requirement of arbitrary CIF files.

The convention is called "x-skeletal connectivity" because the *skeletons* of each object are connected. The x-skeleton of an object is the object formed by shrinking it uniformly by x. Figure 7.1.7 shows an example of a wire connecting to a box; λ-skeletons are shown in solid lines and outlines are dashed. The λ-skeleton of a wire 2λ wide is simply a line (or a connected set of line segments). The λ-skeleton of a box is another box whose center is identical with the first, but whose width and length are diminished by 2λ.

The advantages of skeletal connectivity arise when geometric processing of the CIF file must be done. For example, if a pattern generator requires that all objects be shrunk slightly before exposure, shrinking a CIF file that uses the λ-skeletal connectivity convention can be done on an object-by-object basis: shrinking by any amount less than λ will preserve all electrical connections. If a file does not use the skeletal connectivity convention, shrinking two abutting objects independently will introduce a gap between them.

Design-rule checkers may also find processing of λ-skeletally-connected files easier. All minimum-width checks can be made simply by checking skeletons of objects. Clearance checks may also be conveniently expressed in terms of skeletons. Since many designs use lots of 2λ wires that have simple (line) skeletons, the checker may be able to substitute simple checks on lines for complex checks on solid objects, and consequently run faster.

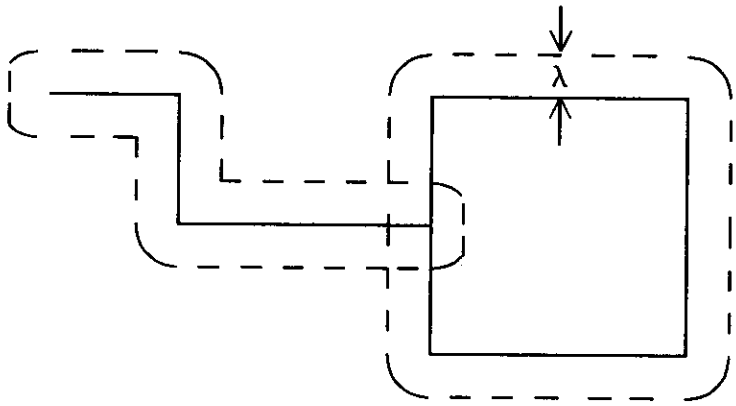


Figure 7.1.7

6. *No-polygon convention.* Files that observe the no-polygon convention do not use the Polygon primitive.

7. *Modest line-length convention.* CIF places no limits on the number of characters in a text line, although many computer systems find processing extremely large lines (or records) impractical. A CIF file that adheres to the modest line-length convention has no lines with more than 132 characters. On record-oriented systems, these files can be accommodated in 132-character fixed-length records. On character-oriented systems, these files have line breaks (carriage-return in the ASCII standard) so that no line is longer than 132 characters. Note that carriage return is a "blank" in CIF syntax, and may not appear at arbitrary points (e.g., in the middle of an integer).

7.1.5 Future Plans for CIF

CIF 2.0, as defined in this chapter, is being used heavily by university communities to transmit experimental designs and student projects to fabrication facilities. Several dozen serious projects have been successfully described in CIF and fabricated. The use of CIF is growing.

Our emphasis at the moment is on gaining experience with CIF 2.0 rather than on extending it. How well does CIF work in practice? What are the problems integrating it into a commercial fabrication environment? Is the definition given above clear and unambiguous? What are the problems with implementing computer programs to generate or parse CIF? Is it feasible for commercially-available design systems to generate CIF output? Does CIF apply equally well to all integrated-circuit technologies? These are the questions we are currently exploring.

When sufficient experience has been accumulated, it is likely that a new version of CIF will be specified. Currently, the suggestions for improvements are:

1. A mandatory "version" command (see *Version comment convention*, section 7.1.4).

2. A sensible convention for defining polygons with "holes." One proposal that has been advanced is to define a polygon with a series of boundary contours, together with a convention about the direction of tracing a boundary (e.g., the interior of the shape is always on the left as the boundary is traversed in the order specified).

3. Better scope rules for symbol definitions. The present symbol machinery has difficulties: DD leads to dangling references; it is difficult, in the general case, to make a "one call" file from an arbitrary CIF file – this makes the assembly of multi-project chips rather difficult. Using nested symbol definitions and some corresponding scope rules will solve these problems.

4. Some mechanism to iterate symbol calls so that regular arrays can be generated easily. Iteration was deliberately omitted from CIF 2.0 because we could not devise a method that avoided machine-dependent computational problems. It is still possible in CIF 2.0 to achieve substantial file compaction when defining arrays by using several layers of symbol (e.g., cell, row, double-row, etc.). In a future extension, we may choose to make the "simple" iterations a part of CIF (e.g., two-

dimensional orthogonal arrays), and omit more complex iterations such as hexagonal arrays. The more complex designs will simply have to call symbols explicitly, rather than using an iterative construct.

This list will doubtless grow as experience with CIF grows. Please help improve CIF by sending us comments and suggestions.

7.2 Ways to Generate CIF

CIF is an interchange format, usually generated by a computer-aided design system (Figure 7.1). There are thus as many ways to generate CIF as there are design systems. This section simply surveys some of the techniques.

7.2.1 Keyboard Interface

For small-scale projects, it is possible to generate a CIF file directly with an interactive text editor. The designer will generally make a detailed drawing of the layout, and read off coordinates from his drawing and enter the appropriate CIF commands one at a time. After the design is entered, it must be check-plotted in some way to verify that the file is in proper CIF format and correctly encodes the layout. In this way, a person can design a small experiment with minimal facilities.

Creating CIF files directly is much easier if the designer has a library of CIF text that defines useful symbols: pads, pad drivers, PLA cells, shift register cells, and so forth. The symbols must be accompanied by documentation that illustrates how to make connections to them, etc. Such a library is a tremendous aid to a beginning designer because it allows him to concentrate on his design ideas rather than on details of pad drivers and the like. The library is, in effect, a primitive design data base. Although CIF is not intended to be used for complex data bases, it can be used very effectively in this crude form.

Direct creation of CIF files can also be made more convenient and less error prone with a suitable preprocessor program. For example, an interactive input program might allow the designer to use relative coordinates, provide suitable default values, permit convenient iteration of cells, and check for obvious errors. It might prompt the user for the next entry and give feedback as to the current symbol definition or current layer. Ideally it would interact with the user in terms of symbolic symbol names and handle the generation of proper symbol numbers internally. (See section 2.2 for a further discussion of such a program.)

7.2.2 Programming Languages

If the designer is also a computer programmer, he may prefer to write a program that, when executed, writes a CIF file. This allows all the facilities offered in programming languages to be used in determining the geometry: arithmetic expressions, conditionals, procedures, etc. Often a cell is used with some minor modifications in many different places of the overall design. This cell could be defined by a procedure, with parameters that govern the modifications. The procedure in turn calls on "primitive" procedures that generate CIF geometric primitives directly. These primitive procedures might also generate a display or check-plot at the same time the CIF file is being constructed. (See [Newman & Sproull 1979] for a discussion of "display procedures." See [Locanthi 1978] for an example of a simple set of procedures for IC design.)

If a fully interactive programming environment is available, the user can change the program and re-execute it very quickly. Most BASIC, APL and LISP systems are interactive in this way, and do not incur the delay of compiling the program before it can be executed. If the execution of the program generates a display, the designer can very quickly modify the program to achieve the design he desires.

An important advantage to this approach is that the programs and procedures created by the designer can be saved for use in subsequent designs. Procedures corresponding to cell designs can be generalized and parameterized to apply in many situations. In effect, the collection of procedures becomes a very powerful design data base.

7.2.3 Interactive Graphical Layout Systems

A common design tool in use in the IC industry today is the interactive graphical layout system, for example the commercially available Calma or Applicon systems. A user views a display of all or part of his design, and requests changes by entering commands with a keyboard and graphical input device such as a tablet. The graphical editing commands are designed to make simple and complex changes naturally. The system should allow the user to insert and move geometrical shapes as easily as moving cardboard pieces on a floor plan. With the same ease the user should be able to change the shape of geometrical features. If the system automatically snaps geometrical shapes to a selectable grid and aligns all shapes parallel to the coordinate axes (or possibly lines at 45 degrees), it frees the designer from the tiring task of paying attention to exact coordinate values.

Present-day graphics systems are not without their shortcomings. Often it is cumbersome to define a collection of items as a symbol and to generate a two-dimensional array of such symbols. A further drawback results from the the limited resolution and size of the screen. It can be difficult to keep track of the overall context in which particular items are being manipulated. For example, it may be hard to select the proper wire out of a large group of wires extending over a significant

distance on the chip.

It is usually a simple matter for these systems to generate CIF, as they retain a precise description of the layout geometry. It may be difficult, however, for them to exploit the CIF symbol mechanism if the layout system has no concept of symbols or if its symbol semantics do not correspond closely with those of CIF.

7.2.4 Standard-cell and Gate-array Systems

Some IC design systems in use today free the designer from most of the details of geometric layout. The design is described in terms of gates or functional blocks and the interconnections among them. The description may be entered with an interactive graphics system, but it contains no geometric information; it is a "logic diagram." The design system processes the logic diagram, drawing on a library of gate or standard-cell geometries, and devises a layout that implements the desired logic diagram.

Ultimately, these systems generate geometric mask information, which can easily be expressed in CIF form. These systems can probably take advantage of CIF's symbol machinery to define standard cells, standard gates, etc.

7.2.5 Silicon Compilers

The ultimate goal is to interact with the computer on a much higher level, where the system designer enters a detailed functional description of his system. From there a sophisticated program, drawing upon the resources of a large collection of properly parameterized subsystems or building blocks in a library, would automatically assemble a possible layout and the corresponding CIF file.

Steps toward this goal are already being taken, see for example [Johannsen 1979] or [Ayres 1979].

7.3 Processing CIF files

The processing of CIF files differs somewhat from that for conventional pattern-generation formats. This section attempts to provide some suggestions and guidelines for implementing the programs that read CIF files.

Any program to read a CIF file will probably have the general structure illustrated in Figure 7.3.1. The *parser* is responsible for reading the CIF text file, parsing commands, assembling arguments, and issuing warning or error messages if the syntax of the CIF file is found to be invalid in some way. The *interpreter* is responsible for retaining symbol definitions, expanding symbol calls, performing transformations, and issuing warnings or errors if difficulties are encountered. Finally, the *output* module receives geometric information from the interpreter for each geometric object in the design, and sends it to the appropriate output device. If they are properly designed, the same parser and interpreter can be used for all CIF processing. Different output modules will be needed to drive different output devices: plotters, pattern-generation equipment, displays, and so forth.

7.3.1 CIF Implementation Guidelines

It is important that programs processing CIF files operate cautiously, maintaining a constant vigilance for mistakes or entries that will not be processed properly. The next three sections provide suggestions corresponding to the three modules: parser, interpreter, and output. It must be remembered that these are suggestions; they are not part of the CIF standard.

These sections describe the construction of program that implement *all* of the CIF standard. Implementations of subsets are often a good idea for getting started (e.g., by restricting CIF coding to use only wires and boxes a somewhat simpler check-plotting program can be built). Any serious use of CIF, however, that accepts CIF files from various sources, must implement the full language.

7.3.1.1 Parser

The parser is responsible for reading the CIF text file, checking syntax, and the like. It should carefully implement the syntax given in section 7.1.1. In addition, several semantic checks should be made:

Numbers. The parser should check to be sure that no number exceeds the representation ability of the computer.

Nonsense arguments. Various arguments to CIF primitives do not make sense, and should generate warning messages. This includes boxes with length or width of zero, round flashes with zero diameter, and direction vectors (either in the box command or in transformations) in which

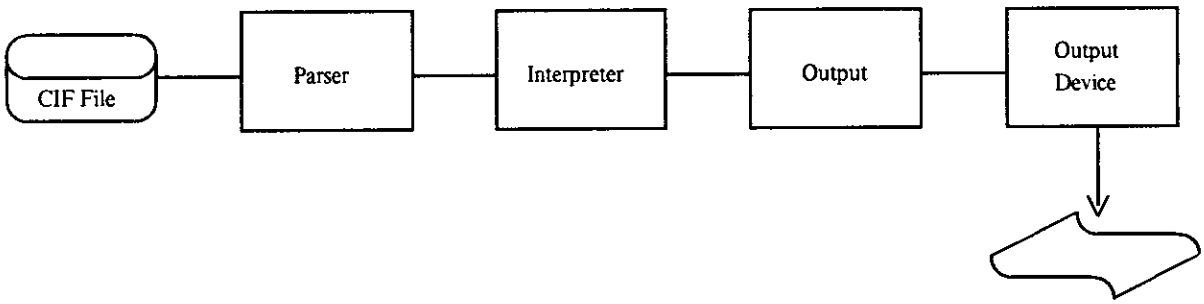


Figure 7.3.1 The Structure of a CIF Processing Program

both elements are zero. A polygon with one or two vertices should generate a warning message; this is almost certainly a mistake. A wire with one point in its path is perfectly legal (it is the same as a round flash at that point, with diameter equal to the wire width), but might deserve a warning message because it is an unusual way to achieve that effect.

Layers. Programs that read CIF will want to check to be sure that layer names used do in fact correspond to fabrication masks being constructed or to plotting conventions if a check-plot is being constructed. However, the file may cite layer names not used in a particular pass over the CIF file. It would be helpful for the program to provide a list of the layer names that it ignored. In this way, a user could spot typographical errors in layer names.

The intention of the layer specification command is to label locally the layer for a particular geometric primitive. It is therefore senseless to specify a box, wire, polygon or flash if no layer has been specified. There are a variety of ways to detect this error. One possibility is to insert the command LZZZZ implicitly at the beginning of the file and as the first command of each symbol definition. Any attempt to generate geometric output on layer ZZZZ will result in an error message.

User extensions. User extensions in CIF provide a means of including non-standard features, which may be installation-dependent. Programs that process CIF should flag and ignore user extension commands that are not implemented. Unfortunately, it is possible to devise user extensions that prevent the CIF file from being interpreted "correctly" by a program that does not implement the same user extensions (see the first example, below). Consequently, there is an informal desire to have similar user extensions. The remainder of this section lists a few machine-independent user extensions that have been found useful. They are described to give readers some feel for the types of features that may be incorporated into CIF. Note that far more than 10 extensions are possible because extensions may have more than one digit (e.g., "889 my own extension;").

Include Files. Format: 0 FileName;

This command has the effect of substituting the contents of FileName for the command. This allows easy merging of a number of separate files to make a project or a multi-project chip. Include File commands may be nested to any level. Programs implementing this command should be careful to flag possible errors, for example if symbol definitions are open across files, or if an End command is encountered within one of the included files. Warning: using this extension makes direct transmission of the CIF file to another site impossible, because it may not provide the same extension. Before transmission, the included file or files must be copied into the file being transmitted.

Comments from CIF Files. Format: 1 "Comment to be typed";

This command provides a mechanism for displaying progress notes to the standard output device (probably the user's terminal) as they are encountered in the CIF file. This is useful

to monitor the progress of the execution of an unobservable output device, e.g. E-beam mask maker.

Text Output on Plots. Format: 2 "TextOnPlot" *transformation*;

This extension cites text to be included on a check-plot to ease identification of pads, parts of the design, or wires. The text is treated as a symbol called with the stated transformations. The first character is located at (0,0) in its coordinate system; the text string extends in the $+x$ direction.

Symbolic Names. Format: 9 SymbolName;

This extension is used to associate a symbolic name with a symbol definition. Example:

```
DS 1 100 1; 9 InputPad;
LNM; (...rest of definition);
DF;
```

End command. The parser might issue a warning message if it finds any non-blank characters following an End command.

7.3.1.2 Interpreter

The job of the interpreter is to retain symbol definitions and to instantiate and transform symbols when they are called.

Transformations. (See also [Newman & Sproull 1979].) When we are expanding a symbol, we need to apply a transformation to the specification of an item in the symbol definition to get the specification into the coordinate system of the chip. There are three sorts of measurements that must be transformed: distances (for widths, lengths), absolute coordinates (for "points" in all primitives) and directions (for boxes). Distances are never changed by a symbol call, because we allow no scaling in the call. Thus a distance requires no transformation.

A point (x,y) given in a symbol definition is transformed to a point (x',y') in the chip coordinate system by a 3x3 transformation matrix T: [x' y' 1] = [x y 1] T
T is itself the product of primitive transformations specified in the call: T = T1 T2 T3, where T1 is a primitive transformation matrix obtained from the first transformation primitive given in the call, T2 from the second, and T3 from the third (of course, there may be fewer or more than 3 primitive transformations specified in the call). These matrices are obtained using the following templates for each kind of primitive transformation:

Tab.	Tn =	1	0	0
		0	1	0
		a	b	1

$$\text{MX.} \quad Tn = \begin{matrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

$$\text{MY.} \quad Tn = \begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

$$\text{Rab.} \quad Tn = \begin{matrix} a/c & b/c & 0 \\ -b/c & a/c & 0 \\ 0 & 0 & 1 \end{matrix} \quad \text{where } c = \text{Sqrt}(a^2 + b^2)$$

Transformation of direction vectors (x y) is slightly different than the transformation of coordinates. We form the vector [x y 0], and transform it by T into the new vector [x' y' 0]. The transformed direction vector is simply (x' y'). Some output devices may require rotations to be specified by angles, rather than direction vectors. Conversion into this form may be delayed until necessary to generate the output format. Then we calculate the angle as ArcTan(y/x), applying care when $x = 0$.

Nested calls require that we combine the transformations already in effect with those specified in the new call. Suppose we are expanding a symbol a, as described above, transforming each coordinate in the symbol to a coordinate on the chip by applying matrix Tac. Now we encounter, in a's definition, a call to b. What is to happen to coordinates specified in b? Clearly, the transformations specified in the call will yield a matrix Tba that will transform coordinates specified in symbol b to the coordinate system used in symbol a. Now these must be transformed by Tac to convert from the system of symbol a to that of the chip. Thus, the full transformation becomes

$$[x' \; y' \; 1] = [x \; y \; 1] \; Tba \; Tac$$

The two matrices may be multiplied together to form one transformation Tbc = (Tba Tac) that can be applied to convert directly from the coordinates in symbol b to the chip. This procedure can be carried to an arbitrary depth of nesting.

To implement transformations, we proceed as follows: we maintain a "current transformation matrix" T, which is initialized to the identity matrix. We use this matrix to transform all coordinates. When we encounter a symbol call, we:

1. "Push" the current transformation and layer name on a stack.
2. Collect the individual primitive transformations specified in the call into the matrices T1, T2, T3 etc.
3. Replace the current transformation T with T1 T2 T3 ... T; i.e., premultiply the existing transformation by the new primitive transformations, in order).
4. Now process the symbol, using the new T matrix.
5. When we have completed the symbol expansion, "pop" the saved matrix and layer name from the stack. This restores the transformation to its state immediately before the call.

Precision in numeric calculations. The numbers given in a CIF file generally undergo substantial processing before being delivered to a pattern generator or a plotter. We must take care that the processing not introduce numerical errors due to insufficiently precise arithmetic in the computer. An argument given below suggests that floating-point arithmetic with 24 or more bits of mantissa will suffice.

First, we note that it is not necessary for CIF to perform calculations with a precision greater than, say, $\lambda/5$. Here we are using λ to denote the characteristic resolution of the lithographic process. Variations in position on the order of $\lambda/5$ will not reproduce.

CIF describes geometry in units of 1/100 micron. This precision should suffice to describe integrated-circuit devices to a precision of $\lambda/5$ at the limit of device operation, roughly .2 μm, $\lambda = .1$ μm. If we imagine the largest design being about 10 centimeters on a side, the largest CIF coordinate would need to be 10^7. To store coordinates precise to 1 part in 10^7 we require 24 bits. An additional sign bit is required because transformations can mirror or translate negative coordinates to positive ones. Happily, these arguments mean that 32-bit floating point numbers, such as are used in the IBM/370 and many other computers, easily suffice for holding CIF coordinates.

These arguments also mean that coordinates may be saved as integers, in units of 1/100 micron. A representation using 25 or more bits (24 plus sign) is required. Note that the scaling by a/b of numbers in a symbol definition may give rise to coordinate numbers with a grain finer than 1/100 micron; these can safely be rounded to units of 1/100 micron.

This discussion ignores any problems with precision in the coordinate systems used to drive plotters, pattern generators, etc. If integer representations are used, conversion from the 1/100th micron system into the system of the output device must be delayed as long as possible to avoid losing precision.

Precision in transformations. Were it not for rotations, the transformation functions would introduce no numerical difficulties. There are two problems introduced by rotations: imprecision in the matrix values (a/c, b/c, etc. in the notation above), and the accumulation of error due to nested symbol calls. A single rotation may introduce some error due to the matrix multiplication (e.g., x' = x*a/c-y*b/c). This calculation introduces approximately 4 round-off errors (the error will either be 1/2 least significant bit if "rounded" arithmetic is used or 1 least significant bit if "chopped" arithmetic is used).

The nesting problem is more dangerous, because the computation of the transformation matrix is *incremental.* If a symbol calls another with a 1-degree rotation, which in turn calls a third with a 1-degree rotation, and so on to a depth of 90 calls, the calculation of the current transformation matrix may have accumulated considerable error. Very roughly, each level of nesting may introduce 4 round-off errors in a coefficient. So to allow nesting to a depth of n we should use arithmetic with a precision of 1 part in $10^7(4n+4)$. Equivalently, we should use floating-point arithmetic with

$26+\log_2 n$ bits in the mantissa. Calls that do not use rotation or that involve only rotations by multiples of 90 degrees (which use only perfectly accurate coefficients 0 and 1, and therefore give rise to perfectly accurate matrix multiplication) do not need to be counted in n. All of this suggests that the extremely cautious interpreter will keep track of symbol nesting depth, and issue a warning if it suspects arithmetic precision is insufficient.

Symbols. There is no sensible way in which a symbol may be invoked recursively (i.e., call itself, either directly or indirectly). Programs that read the intermediate form might check that no recursion occurs. This can be achieved by retaining a single flag with each symbol to indicate whether the symbol is currently being instantiated; the flags are initialized to "false." When a symbol is about to be instantiated, we check the flag; if it is "true," we have detected recursion, and so print an error message and do not perform the call. Otherwise, we mark the flag "true," instantiate the symbol as specified, and mark the flag "false" when the instantiation is complete.

7.3.1.3 Output

The most difficult problem faced in the output module is that of *device independence*: how is the geometry specified in the CIF file to be generated on a particular output device with sufficient precision? A complete discussion of methods used to drive different output devices is beyond the scope of this document. This section attempts instead to discuss the device-independence problem in general terms.

Driving any device will require choosing techniques for generating the geometry CIF specifies on that device. These choices are simple for devices such as pen plotters, which can trace arbitrary shapes with modest precision. By contrast, a pattern generator may allow only rectangular "flash" exposures of limited size. Moreover, the precision with which the CIF geometry is reproduced on the reticle is extremely important.

It is often impossible to guarantee that the geometry will be reproduced *exactly*; instead we require that the software and pattern generator exert their "best efforts" to conform to the CIF. Once again, the parameter λ enters: the design rules allow us some leeway in making mask patterns. The exact details of how much error can be tolerated will depend on the entire fabrication process, not just the pattern-generation step.

The fact that pattern generators cannot reproduce CIF exactly leads to a requirement for *approximations*. Examples of the kinds of approximations that may be necessary are:

1. Round flashes might be approximated with octagons, or with two overlapping square flashes rotated 45 degrees (Figure 7.3.2). Another approximation might use more flashes at smaller rotational increments.

2. Wires cannot be constructed exactly on output devices that cannot construct circles. One possibility is to use the round flash approximations described above. Another is to "square off" the ends of wires by extending them by the half-width of the wire, and thus generate only box shapes

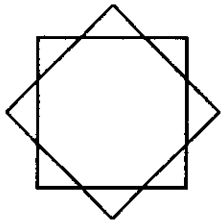


Figure 7.3.2 Approximation of a Round Flash

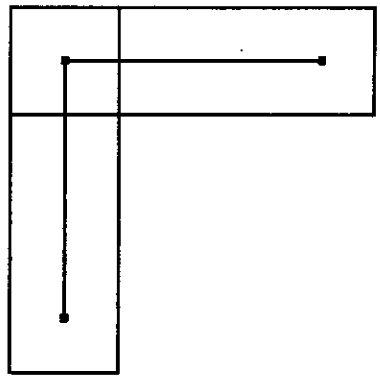
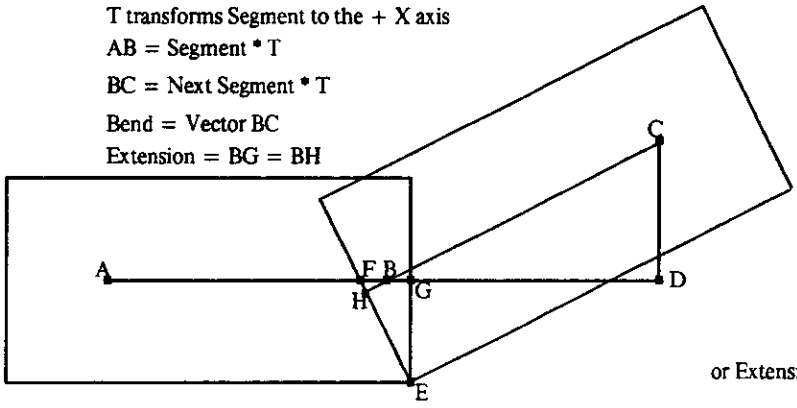


Figure 7.3.3a Approximation of a Wire



Similar triangles BCD, EFG, BFH
BC:CD:DB::EF:FG:GE::BF:FH:HB

FG = FB + BG
= BH * (BC/DB) + BG
= (1 + BC/DB) * BG

BG = FG/(1 + BC/DB)
= GE * (CD/DB) / (1 + BC/DB)
= GE * (CD/(DB + BC))

or Extension = w/2 * Y [Bend] / (X[Bend]] + LENGTH[Bend])

Figure 7.3.3b Converting Wires to Boxes

(Figure 7.3.3a). This approximation works nicely for segments that meet at right angles, but may cause problems if wires or wire segments are connected at arbitrary angles. The technique can be improved by adjusting wire lengths and positions (Figure 7.3.3b). The width of the boxes is the same as the width of the wire, but the length of the box is adjusted to reduce unfilled wedges are overlapping "ears." An algorithm for constructing boxes in this way from a wire is given below. If the wire is specified in a symbol definition, the approximation need be computed only once, and can then be used each time the symbol is instantiated.

The following algorithm for decomposing wires into boxes was developed by Carver Mead, and first implemented at Caltech by Ron Ayres; it was was further modified to be consistent with the use of direction vectors, to allow more general path lengths, and to avoid use of trigonometric functions. Note that this decomposition covers more area than the locus of points within w/2 of the path for small angles of bend, but less area for sufficiently sharp bends; in particular, if a path bends by 180 degrees (reverses) it will have no extension past the point of reversal (it is missing a full semicircle). Other decompositions are possible, and may better approximate the correct shape.

```
Let the wire consist of a path of n points p1,...,pn.
Let w represent the width of the wire.
    IF n = 1 THEN
        {MAKEFLASH[Diameter ← w; Center ← p1]: "single-point gets a flash";
        DONE;};
    i ← 1;
    OldExtension ← w/2; "initial end of wire"
    Segment ← p2 - p1; "p1 and p2 are points in path, Segment is a vector (a point)"
    "LoopConditions:"
    FOR pi, pi+1 in path UNTIL pi+1 is last DO
        "calculate the box for the segment from pi to pi+1:"
        IF pi+1 is last THEN
            { Extension ← w/2; "final end of wire" }
         ELSE
            { "compute Extension for intermediate point:"
            NextSegment ← pi+2 - pi+1; "next vector in path"
            T ← MATRIX[ X[Segment],  -Y[Segment],
                        Y[Segment],   X[Segment] ];
            "T transforms Segment to +x axis."
            Bend ← MULTIPLY[ NextSegment, T ]; "relative direction vector"
            "if Bend is (0 0), delete pi+1, reduce n, and start over"
            Extension ← w/2 * ( ABS[Y[Bend]] /( LENGTH[Bend] + ABS[X[Bend]] ) ); };
        MAKEBOX[
            Length    ← LENGTH[Segment] + Extension + OldExtension;
            Width     ← w;
            Center    ← ( pi + pi+1 )/2 +( Segment / LENGTH[Segment] ) *(Extension - OldExtension)/2;
            Direction ← Segment; "careful, may be zero vector" ];
        i ← i + 1;
        OldExtension ← Extension;
        Segment ← NextSegment; "next vector in path"
        ENDLOOP;
    DONE;
```

This is only one of many possible algorithms. When a particular approximation for wires is chosen, the ideal concept of a wire with semi-circular ends should always be kept in mind.

Polygons. Polygons may need to be approximated by boxes. In some cases, the approximation can be exact or nearly so (Figure 7.3.4a). In others, it will be impossible to avoid introducing "ears" or unfilled wedges (Figure 7.3.4b). Alas, we have no polygon-approximation algorithm to recommend.

The designer of the output module must be careful to handle self-intersecting polygons properly. If the implementation cannot deal with these polygons, the program should at least detect self-intersecting polygons and issue appropriate error messages.

The correct processing of self-intersecting polygons may depend on the type of output device. A geometric transformation can be applied to generate a new set of polygons that are not self-intersecting [Sutherland 1978]; these can then be approximated with the same technique as normal polygons. For line-by-line raster-scanned devices the following approach will give the proper result. With each polygon edge, an orientation is maintained (upward-moving or downward-moving), given by the original ordering of the points in the descriptive path. Moving across a scan line, a counter keeps track of how many intersections with the two types of edges have been encountered: the counter is incremented when an upward-moving edge is crossed, and decremented when a downward-moving edge is crossed. Whenever the the counter is non-zero, the scan line is inside the polygon. In these regions of the scan-line, the image of the polygon is generated.

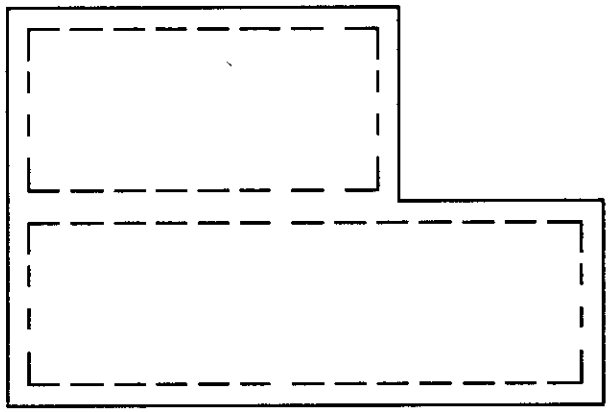
7.3.2 A Program for Processing CIF

Once an IC design has been expressed in CIF, a number of different programs are needed to create different representations. An obvious need is for a program to convert CIF into a form suitable for generating masks. Other programs may create checkplots from CIF, a file listing design rule violations, or an input file for a simulator. All of these programs require a CIF file as input and produce output in some other form. Accordingly, the "front ends" of these programs are likely to be very similar and could therefore be shared.

The remainder of this section details a modularization of one program that inputs CIF and plots it. The document is intended to provide guidelines for one reasonable structure for such a program, which is not optimized for any particular output device. The reader should be familiar with the syntax and semantics of CIF, and have given the problem of processing CIF some thought before trying to fully understand the program structure. The procedure declarations are written in a Pascal-like language, the main difference being that routines may return arbitrary structures. *All parameters are passed by value.*

The system is composed of a Parser, an Interpreter, and an Output stage. The Parser reads characters from an input file, checks syntax and generates well-defined, parameterized procedure calls on routines in the Interpreter. These calls closely reflect the structure of CIF; for each input statement, one command to the Interpreter is generated. The Interpreter takes the command and either generates a call, such as *Output a wire...*, to the Output module or saves it (in some internal

(a)



(b)

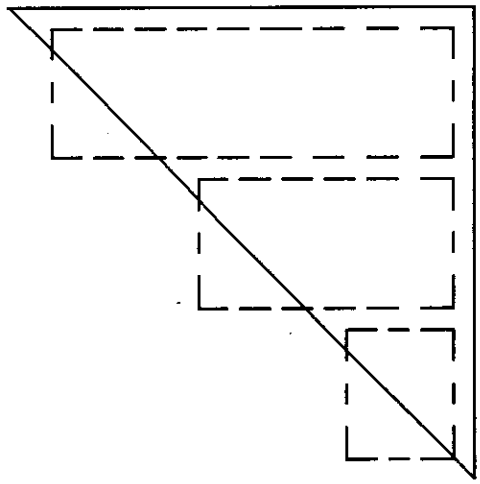


Figure 7.3.4 Approximations of Polygons

form) for later recall. The Output module produces an output file, plot, or display containing the result of output commands. Together the Parser and the Interpreter comprise a front end that may be combined with individual Output stages for any of a number of devices. The Parser and the Interpreter can be written with about one man-month of effort. Depending on the complexity of the output device, new Output stages may require from one man-week to one man-month of effort to implement.

The section on each major component contains suggestions on how it might be implemented. It bears repeating that the implementation suggested is not intended to be particularly efficient for all output devices. Raster scan devices, for example, may require output commands to be sorted in some order. This affects the internal structure of the Interpreter, and may dictate major changes in the way the Interpreter is implemented. Nevertheless, the routines that the Interpreter provides to the Parser should have the same input/output behavior as before.

Several modules are needed by all three phases of the CIF processing program; these modules provide services for reporting errors, typing messages on the user's terminal, and other utility functions. Each module contains a routine to initialize the module that must be called before any of the others, and a routine called to terminate the interaction with the module. There must also be a main program that initializes each module, interacts with the user to obtain input parameters (e.g. file names), contains a loop to scan the next CIF statement, and cleans up.

Data Type Definitions. A number of data type definitions are required by various modules and are included here. It is suggested that procedures that support these data types (such as routines to add a Point to a Path) be packaged together in one module with the TYPE declarations to form suitable data abstractions. The types used are only guidelines, as each implementor has his/her own preferences for data structures.

```
Boolean InitTypes()

Boolean FinishTypes()

Type Point =
        Record
        Integer X, Y;
        End;

Type LinkedPoint =
        Record
        Point Value;
        ↑LinkedPoint Next;
        End;

Type PathRecord =
        Record
        ↑LinkedPoint First, Last;
        Integer Length;
        End;

Type Path = ↑PathRecord;

Path AllocatePath()
        Paths are queues of Points; used to specify the perimeter of polygons and the trail that wires follow.

FreePath(Path Name)
        Release all of the storage held by Name.
```

```
AppendPoint(Path Name, Point Value)
        Adds a point onto the end of Name.

Boolean, Point RemovePoint(Path Name)
        Returns True and the next point on Name or False and garbage if Name is an empty path.

Integer PathLength(Path Name)
        Returns the number of points remaining on Name.
```

TLists are queues of transformation commands (the commands may be represented in any convenient manner, say as strings -- "R 1 1" -- or perhaps as triples of integers -- "0 1 1" -- where the "0" is interpreted to mean "Rotate"). Used to pass a list of CIF transformations to the Interpreter, which ultimately converts the list into a transformation matrix.

```
Type TType = {Mirror, Translate, Rotate};

Type TEntry =
        Record
        Select TType: ThisOne FROM
        Mirror => [Coords: {X,Y}],
        Translate => [Integer X, Y],
        Rotate => [Integer XRot, YRot],
        End;

Type LinkedTEntry =
        Record
        TEntry Value;
        ↑LinkedTEntry Next;
        End;

Type TRecord =
        Record
        ↑LinkedTEntry First, Last;
        Integer Length;
        End;

Type TList = ↑TRecord;

TList AllocateTList()

FreeTList(TList Name)

AppendTList(TList Name, TEntry Value)

Boolean, TEntry RemoveTList(TList Name)

Integer TListLength(TList Name)
```

Error Reporter. Each module reports errors that it encounters through a call to "Report" in the Error Reporter. This module takes care of identifying the input file context of the error and reporting and logging errors in a consistent manner.

```
Type Error = {FatalSyntax, FatalSemantic, FatalOutput, FatalInternal, Advisory, Fatal, Other};

Boolean InitError()
        Initializes the Error module, returns False on error.

Boolean FinishError()

Report(String Message; Error Kind)
        Indicates the current context (by a call to Identify() in the Input Module) and prints Message (a call
        to SendMessage() in the Output Module), keeps a count of each type of error. Advisory errors are
        preceded by a "Warning" tag.

Array of Integer ErrorSummary()
        Returns the number of each type of error encountered so far (the array is as long as there are
        different kinds of errors).
```

Input. All Input functions are carried out by this module, in addition, it keeps track of the input context and provides a means of displaying it (for use by the Error Reporter).

Boolean InitInput()
Initializes the Input module, returns False on error.

Boolean FinishInput()
Cleans up the Input operations (e.g. closes open files), returns False on error.

Boolean InFromFile(String Filename)
Fixes things so that input characters come from FileName (i.e. the CIF file).

Character GetChar()
Removes and returns the next character from the currently open file. Succesive calls to Getchar return successive characters from the file. Returns EOF when the end of file is reached.

Character Peek()
Returns the next character from the currently open file without removing it from the input stream (providing one character look ahead).

Boolean EndOfFile()
Returns True if the end of the input file has been reached.

Character Flush(Character BreakChar)
Tosses out characters up to and including the first BreakChar, returns the character it stopped on (either BreakChar or EOF). Useful in error recovery.

Identify()
Identifies the context within the input file by line number, contents, or whatever. Calls SendMessage in the Output module to have the line printed on the terminal.

7.3.2.1 Parser

The Parser is responsible for reading the input file and generating calls to the Interpreter. Its primary responsibility is syntax checking (and perhaps some limited semantic checking).

Type CommandType = {Wire, DefineStart, DefineEnd, DeleteDef, CallSymbol, Layer, Flash, Polygon, Box, End, Comment, NullCommand, UserCommand, SyntaxError, SemanticError};

These are the codes returned by ScanStatement; there is one for each type of command possible (cf. CIF syntax).

Boolean InitParser()

Boolean FinishParser()

CommandType ScanStatement()
Scans the statement (= {*blank*} [*command*] *semi*) beginning at the current character, munching chars up to and including the closing *semi*. Once a syntactically valid statement has been scanned a procedure in the Interpreter is called. Returns a code for the type of statement scanned.

Implementation. One simple way to do syntax analysis is to write a procedure to parse each of the nonterminals (e.g. *semi*, *sep*, *blank*) in the CIF syntax. Characters from the input file are obtained using the procedures in the Input module, and the user is notified of errors through the Error Reporter. A CASE statement on the next character in the input file suffices to decide which type of statement is expected; each arm of the CASE statement is a sequence of calls on the routines that parse nonterminals. Error recovery can be relatively crude inititally, for example an error might cause the Parser to ignore characters through the next *semi*. Front ends that are intended to process CIF that was generated by hand should be especially careful to catch machine dependent errors (e.g. word length limitations). The function of the Parser is essentially limited to

syntax analysis; it should do as little interpretation as possible.

7.3.2.2 Interpreter

The Interpreter receives commands from the Parser as each CIF command is parsed. It keeps track of the current state (whether or not a definition is in progress, the current layer, etc.) and based on that information makes the appropriate calls on the Output module. Each call on ICallSymbol, IWire, IFlash, IPolygon, or IBox in the Interpreter will either cause an output command to be generated, or will be stored (because it is part of a symbol definition). Those items that are stored will be output each time the symbol that they are part of is called outside of a symbol definition.

Boolean InitInterpreter()
Initializes the Interpreter, returns False on error.

Boolean FinishInterpreter()
Cleans up after the Interpreter, returns False on error.

IDefineStart(Integer SymbolNumber, Multiplier, Divisor)
Get things set up so that items are stored (probably in some compiled form) as part of this symbol definition. IDefineStart will have to create a symbol table entry for SymbolNumber after making sure that no other symbols are currently being defined. If SymbolNumber is already defined, a warning message should be given. Multiplier and Divisor are used to scale incoming points and dimensions.

IDefineEnd()
Finish up the current symbol.

IDeleteDef(Integer NSym)
Delete all symbols numbered NSym and above.

ICallSymbol(Integer SymbolNumber; TList A)
Make a call to SymbolNumber with the transformations given by list A. If a symbol definition is in progress the call will be stored for later recall. Otherwise the Interpreter must retrieve the primitive objects from SymbolNumber and generate a call on the Output module for each.

ILayer(String LayerName)
Switch to layer LayerName.

IWire(Integer Width; Path A)

IFlash(Integer Diameter; Point Center)

IPolygon(Path A)

IBox(Integer Length, Width; Point Center; Integer XRotation, YRotation)
A call on any of these routines will cause one of two things to happen: if a symbol definition is in progress, some compiled form of the command will be saved in the internal data structures within the Interpreter. If there is no symbol definition in progress, a call will be made on Output routines.

IComment(String Contents)
The interpretation of comments will vary with the application of the CIF processing program. Where some form of checkplotting is being done, comments will likely be ignored.

IUserCommand(Integer Command; String UserText)
The meaning of user commands will also vary, those user commands not recognized by the interpreter should be passed on to the Output module.

IEnd()
An End command has been scanned.

Implementation. The Interpreter contains all of the symbol table routines necessary to maintain the symbol number / symbol definition correspondence, a storage manager that allows

arbitrarily large symbol definitions to be stored, and also does semantic checking for nonsense arguments or other problems (e.g. output to layer ZZZZ). For every command parsed the Interpreter takes one of four actions:

1. *It passes the command along to the Output module.* This happens when a primitive (Wire, Box, Polygon, Flash) occurs outside of symbol definition.

2. *It stores the command in the current symbol definition.* This happens when a Call or a primitive is encountered within a symbol definition. Symbol numbers may be hashed into a table that holds a pointer to the first and last primitives in the definition. The primitives can be stored as a linked list; primitives are added to a symbol by tacking them on after the last entry. Each type of primitive requires that different information be stored with it. Calls need an incremental transformation matrix (see GetLocal below) and the symbol number called. Boxes require a length, width, center and layer.

3. *It changes some internal state information (e.g. DF causes the current symbol to be closed) and no output action is taken.* This case includes layer changes, DS, DF, DD.

4. *It recalls a symbol definition, making a number of calls on the Output module.* In calling a symbol, the Interpreter must set up the transformation stack correctly. For example, the command "C 114 T 10000,0 M X" is encountered. The Interpreter does a SaveTransformation() to begin a new frame of reference. It then calls Translate(10000,0) and Mirror(True) to set up the incremental transformations. It looks up symbol 114, obtaining a pointer to the first primitive in the definition; at the same time, symbol 114 is marked as "being expanded." It then makes a call on the Output module for each piece of primitive geometry. The Output module will make calls on TransformPoint in order to interpret coordinates in the current context. If a symbol call is encountered in the symbol being expanded, the entire process is nested one level deeper. When the end of the symbol definition is reached, the Interpreter resets the expansion flag, and calls RestoreTransformation() to reset the frame of reference.

Transformations. The Transformation module contains a stack of transformation matrices; the top matrix is the "current" matrix. A new transformation system is created by a call to SaveTransformation, which pushes a new matrix (initialized to an identity matrix) onto the stack. Subsequent calls to Mirror, Translate, and Rotate modify the top matrix. On the first call to TransformPoint, the top matrix is postmultiplied by the matrix that is underneath it on the stack and replaced by the result. This represents the new frame of reference, which is the previous frame with the new transformations applied; the point passed to TransformPoint is postmultiplied by this new matrix to give the final point. Further calls on TransformPoint use this matrix, until the context is changed; it is illegal to issue more transformation commands without changing the context (by a call on SaveTranformation or RestoreTransformation). The previous context is

restored by a call to RestoreTransformation, which pops the stack.

```
Type TransformationMatrix =
        Record
        Real a11,a12,a21,a22,a31,a32,a33;
        Boolean IdentityMatrix;
        End;

Boolean InitTransformation()

Boolean FinishTransformation()

Rotate(Integer XRotate, YRotate)
        Builds the rotation into the current transformation matrix.

Translate(Integer XTrans, YTrans)
        Builds the translation into the current transformation matrix.

Mirror(Boolean XCoord)
        Builds the mirroring (either the X coordinates or the Y coordinates) into the current transformation
        matrix.

Point TransformPoint(Point A)
        Replaces the top matrix with the product of the top matrix and the previous matrix (only if the two
        have not been multiplied yet, i.e. on the first call to TransformPoint) and returns a point as
        transformed by the current (top) transformation matrix.

SaveTransformation()
        Pushes a new identity transformation matrix onto the stack, thereby starting a new relative coordinate
        system.

RestoreTransformation()
        Pop the transformation stack, return to previous coordinate system.

TransformationMatrix GetLocal()
        Returns the top matrix on the stack, which represents the incremental transformations applied since
        the last SaveTransformation. Useful for saving transformations for a symbol call.

ApplyLocal(TransformationMatrix T)
        Apply T to the top of stack, i.e. top ← top x T. Used to set up the context for a symbol call.
```

7.3.2.3 Output

The Output module produces the output (e.g. raster scan bit maps, CalComp plot commands, MEBES trapezoids and rectangles, or some intermediate form that requires further processing) that is the end result. It contains routines for each of the primitive geometric constructs defined in CIF. The CIF geometric primitives have not been subdivided (to the level of rectangles, for instance) so that redundant work is eliminated and so that the advantages of each output device may be fully utilized. For example, while a vector plotter would like to have rectangles expressed as four corners, a PG machine would require that a four-corner representation be converted to a length, width, and angle. By forcing the Output stage to do the conversion from CIF primitive to whatever form is best for the output device, such convert/unconvert problems are avoided.

Coordinates passed to the Output stage are always interpreted within the current transformation context, that is, the final coordinates are obtained by a calls to "TransformPoint" in the Transformation Module (which is part of the Interpreter, see above). The Output module may modify the transformation context (through other calls to the Transformation Module) in order to implement transformations that it may require, but should reset the context to the state it was in prior to the output call before returning.

Boolean InitOutput()
Initializes the Output module, returns False on error.

Boolean FinishOutput()
Cleans up, returns False on error.

Boolean OutputToFile(String FileName)
Fixes things so that the output goes to FileName (may not be needed in some cases).

Boolean MessagesToFile(String FileName)
Routes messages sent by SendMessage to FileName instead of the terminal, useful for keeping a log of messages printed to the user.

SendMessage(String Text)
Prints Text wherever messages get printed, probably the terminal, intended for reporting error messages somewhere.

OutputWire(String Layer; Integer Width; Path A)

OutputPolygon(String Layer; Path A)

OutputBox(String Layer; Integer Length, Width; Point Center; Integer XRotation, YRotation)

OutputFlash(String Layer; Integer Diameter; Point Center)
These commands cause each of the primitive geometric items to be output.

OutputUserCommand(Integer Command; String UserText)
The meaning of user commands will vary, those user commands not recognized by an installation should be flagged with warning messages.

7.4 A Final Note

It is our ambition to refine and improve CIF and its description in this chapter. Comments and questions about CIF or about its presentation here are most welcome. Please send them to:

Robert F. Sproull
Computer Science Dept.
Carnegie-Mellon University
Pittsburgh, Pa. 15213

or to ARPANET address:

Sproull@CMUA

References

[Ayres 1979]
R. Ayres, "Silicon Compilation — A Hierarchical Use of PLAs", *Proc. Caltech Conf. on Very Large Scale Integration*, January 1979.

[Johannsen 1979]
D. Johannsen, "Bristle Blocks: A Silicon Compiler", *Proc. Caltech Conf. on Very Large Scale Integration*, January 1979.

[Locanthi 1978]
B. Locanthi, "LAP: A Simula Package for IC Layout", Caltech Computer Science Display File, July 1978.

[Mead & Conway 1980]
C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA., 1980.

[Newman & Sproull 1979]
W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, 2nd Ed., McGraw-Hill, 1979.

[Sutherland 1978]
I. E. Sutherland, "The Polygon Package," Caltech Computer Science Display File, March 1978.

[Wirth 1977]
N. Wirth, "What Can We Do About the Unnecessary Diversity of Notations for Syntactic Definitions?", *Communications of the ACM*, November 1977.

Appendix A. Optical and E-Beam Mask Specifications

Date: August 23, 1978
Requestor: Bob Hon 494-4324
Project Name: PARC-MPC

Reticle Specifications

NOTE: Please return all pattern generator output to the customer.

Number of Reticle Sets: 1
Number of Reticles per Set: 5
Maximum Pattern Dimensions (outside scribes) X: 9348μ. Y: 6324μ
Step and Repeat distance X: 9288μ Y: 6264μ
Reticle Magnification: 10x
Parity Marks on PG Tape? No, please add as needed.
Fiducials on PG Tape? No, please add as needed.
Blow Backs? Yes (color) **Magnification:** 150x **Number of Sets:** 2
Please use the following colors:

DIF	-	GREEN
IMP	-	YELLOW
POL	-	RED
CUT	-	BLACK
MET	-	BLUE

Black and Clears: Yes (8½" x 11") **Number of Sets:** 2

Mask Specifications

Working Plate Material: AR Chrome
Working Plate Size: 4"
Pattern Size: 3"
Number of Working Plates: 2 per level, except 3 of CUT level
Master Plate Defect Density: Standard
Working Plate Specs: (each reticle has the process step name in upper left corner)

Process Step	Number of Flashes	WP Field	CD digitized width	Tolerance
DIF	41783	OPAQUE	6.0μ	0.5μ
IMP	3741	CLEAR	12.0μ	0.5μ
POL	54184	OPAQUE	6.0μ	0.5μ
CUT	18331	CLEAR	6.0μ	0.5μ
MET	29764	OPAQUE	12.0μ	0.5μ

Working Plate Labels:

Process Step	Working Plate Label (19 characters max)
DIF	PARCMPC 878 DIF
IMP	PARCMPC 878 IMP
POL	PARCMPC 878 POL
CUT	PARCMPC 878 CUT
MET	PARCMPC 878 MET

PG Tape:

The files are on the PG tape in Mann 3000 format. The order is: IMP, DIF, POL, CUT, MET. A copy of the directory follows.

Directory (file 0):

WHOLECHIPZ0C00010081WHOLECHIPZ1C00020865WHOLECHIPZ2C00031095WHOLECHIPZ3C00040342WHOLECHIPZ4C00050636$

Overall View of the MET layer

Critical Dimensions are in this area (see closeup on next page)

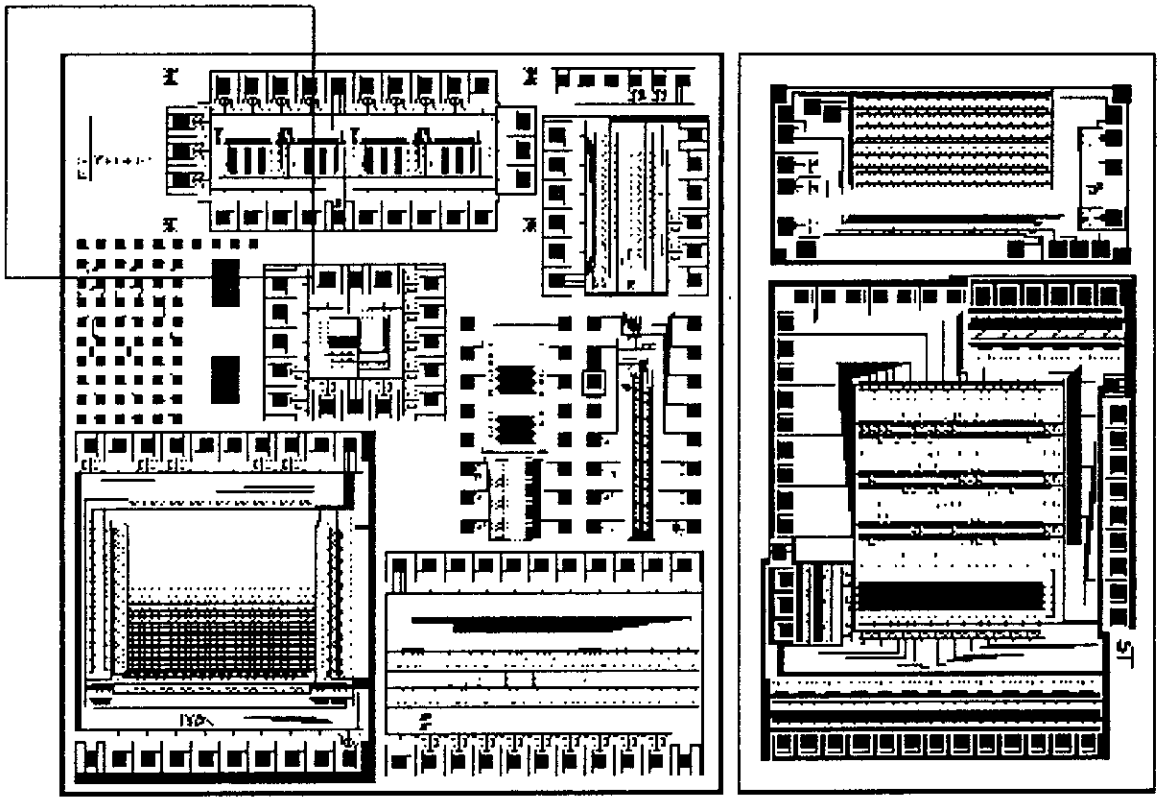


Figure A.1 Diagram of the MET Layer Showing CD Location

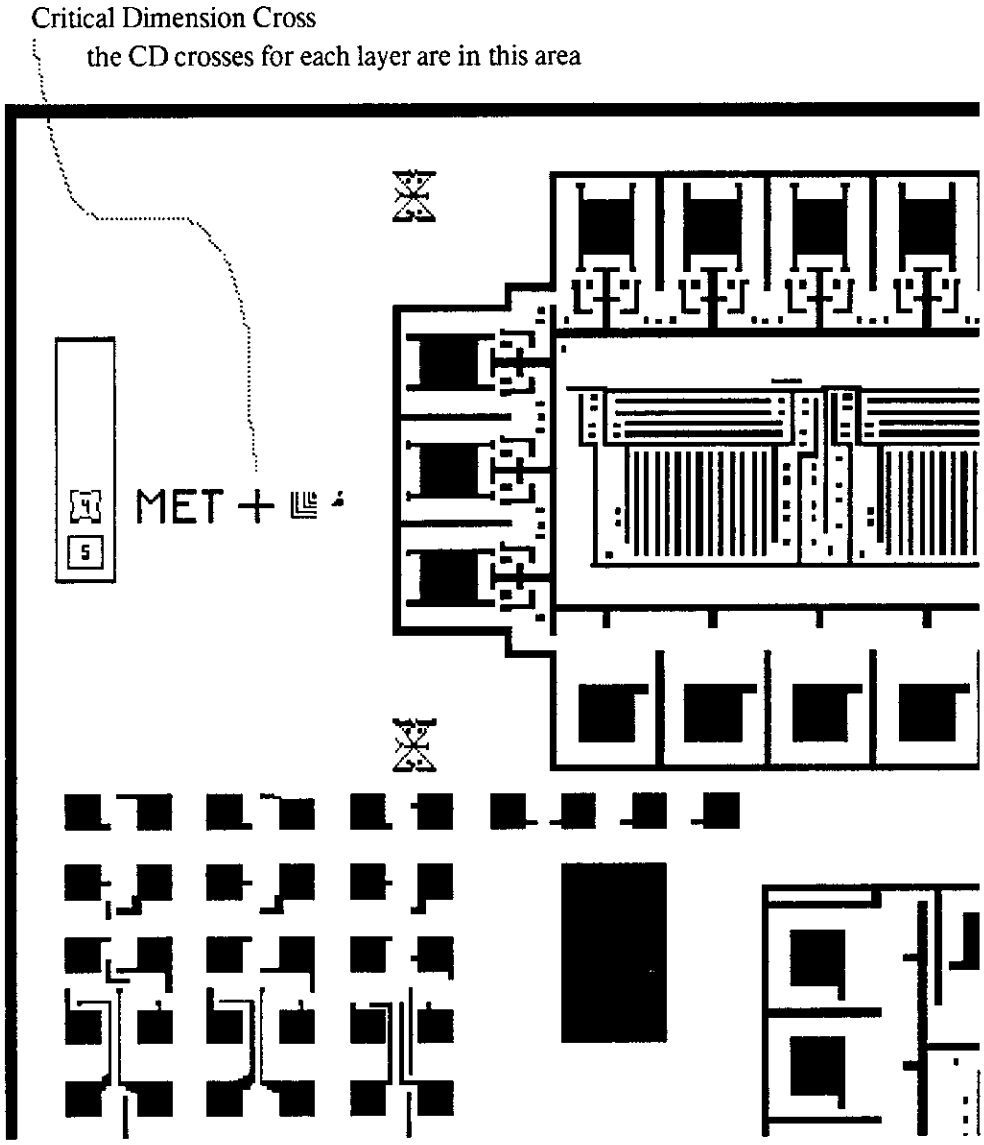


Figure A.2 Close-up of the CD on the MET Layer

Date: May 14, 1979
Ordered by: Bob Hon 494-4364
Project Name: MPC479

MEBES Master Plate Specifications

Maximum Pattern Dimensions (outside scribes) **X:** 6604μ **Y:** 6604μ
Step and Repeat distance X: 6604μ **Y:** 6604μ
E-Beam Spot Size: 0.50μ
Master Plate Defect Density: Standard
Number of Plates per Set: 6
Plate Material: AR Chrome, L.E. 30
Plate Size: 4" x 4" x .090"
Pattern Size: 3", Round
BlowBacks: *Versatec* ***Magnification:*** *127x* ***Number of Sets:*** *1*

Device Label (to appear on each plate): "PARC MPC479 A"
Mask Labels:

Process Step	Mask Label
DIF	XEROX DIF 1A
IMP	XEROX IMP 2A
POL	XEROX POL 4A
CUT	XEROX CUT 5A
MET	XEROX MET 6A
PAD	XEROX PAD 7A

Pattern Specifications: There are three different project dies (named BARBELL, SQUIGGLE, SQUARE) plus one test pattern die (TP9). All have identical dimensions (6604μ x 6604μ). The three project dies are identified by a shape in the upper right hand corner. See the diagrams on the following pages for the location of the CD's and the pattern arrangement on the masks.

PLEASE adjust the patterns so that the text on the masks is WRONG reading when viewed from the chrome side.

PG Tapes: Each project die is on a different mag tape in Mann 3000 metric format (digitized at 10x). The order is: DIF, IMP, POL, CUT, MET, PAD. A copy of the directory for each pattern is given below. The test pattern (TP9) is being sent directly to Micro Mask from Xerox-MEC as an **MEBES Format** mag tape.

Layers which require line width stretching are marked by a * below.

BARBELL

Tape Directory:

```
BARBELLXXDIF00010459BARBELLXXIMP00020046BARBELLXXPOL00030457BARBELLXXCUT0
0040338BARBELLXXMET00050358BARBELLXXPAD00060017$
```

Process Step	Number of Flashes	Plate Field	CD width as DRAWN	CD width on MASK	Tolerance
DIF	17024	OPAQUE	6.0μ	8.0μ *	+/- 0.25μ
IMP	1840	CLEAR	6.0μ	6.0μ	+/- 0.25μ
POL	16377	CLEAR	6.0μ	6.5μ *	+/- 0.25μ
CUT	15653	CLEAR	6.0μ	6.5μ *	+/- 0.25μ
MET	13409	CLEAR	6.0μ	7.0μ *	+/- 0.25μ
PAD	687	CLEAR	6.0μ	6.0μ	+/- 0.25μ

SQUIGGLE

Tape Directory:

```
SQUIGGLEXDIF00010534SQUIGGLEXIMP00020057SQUIGGLEXPOL00030531SQUIGGLEXCUT0
0040371SQUIGGLEXMET00050410SQUIGGLEXPAD00060017$
```

Process Step	Number of Flashes	Plate Field	CD width as DRAWN	CD width on MASK	Tolerance
DIF	20247	OPAQUE	6.0μ	8.0μ *	+/- 0.25μ
IMP	2293	CLEAR	6.0μ	6.0μ	+/- 0.25μ
POL	19855	CLEAR	6.0μ	6.5μ *	+/- 0.25μ
CUT	17251	CLEAR	6.0μ	6.5μ *	+/- 0.25μ
MET	15850	CLEAR	6.0μ	7.0μ *	+/- 0.25μ
PAD	673	CLEAR	6.0μ	6.0μ	+/- 0.25μ

SQUARE

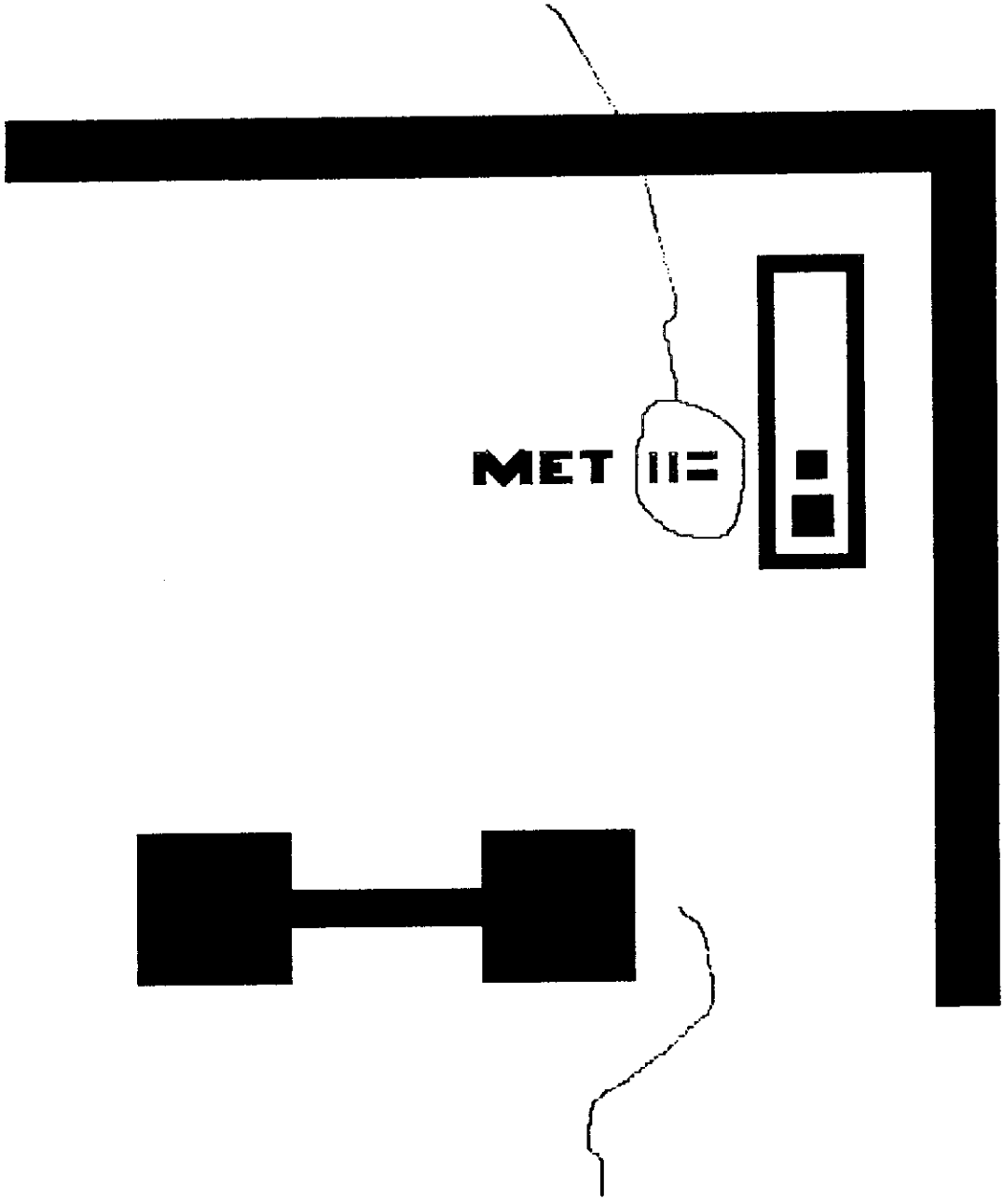
Tape Directory:

```
SQUAREXXXDIF00010753SQUAREXXXIMP00020058SQUAREXXXPOL00030780SQUAREXXXCUT0
0040486SQUAREXXXMET00050565SQUAREXXXPAD00060017$
```

Process Step	Number of Flashes	Plate Field	CD width as DRAWN	CD width on MASK	Tolerance
DIF	36278	OPAQUE	6.0μ	8.0μ *	+/- 0.25μ
IMP	2587	CLEAR	6.0μ	6.0μ	+/- 0.25μ
POL	37461	CLEAR	6.0μ	6.5μ *	+/- 0.25μ
CUT	24314	CLEAR	6.0μ	6.5μ *	+/- 0.25μ
MET	25842	CLEAR	6.0μ	7.0μ *	+/- 0.25μ
PAD	669	CLEAR	6.0μ	6.0μ	+/- 0.25μ

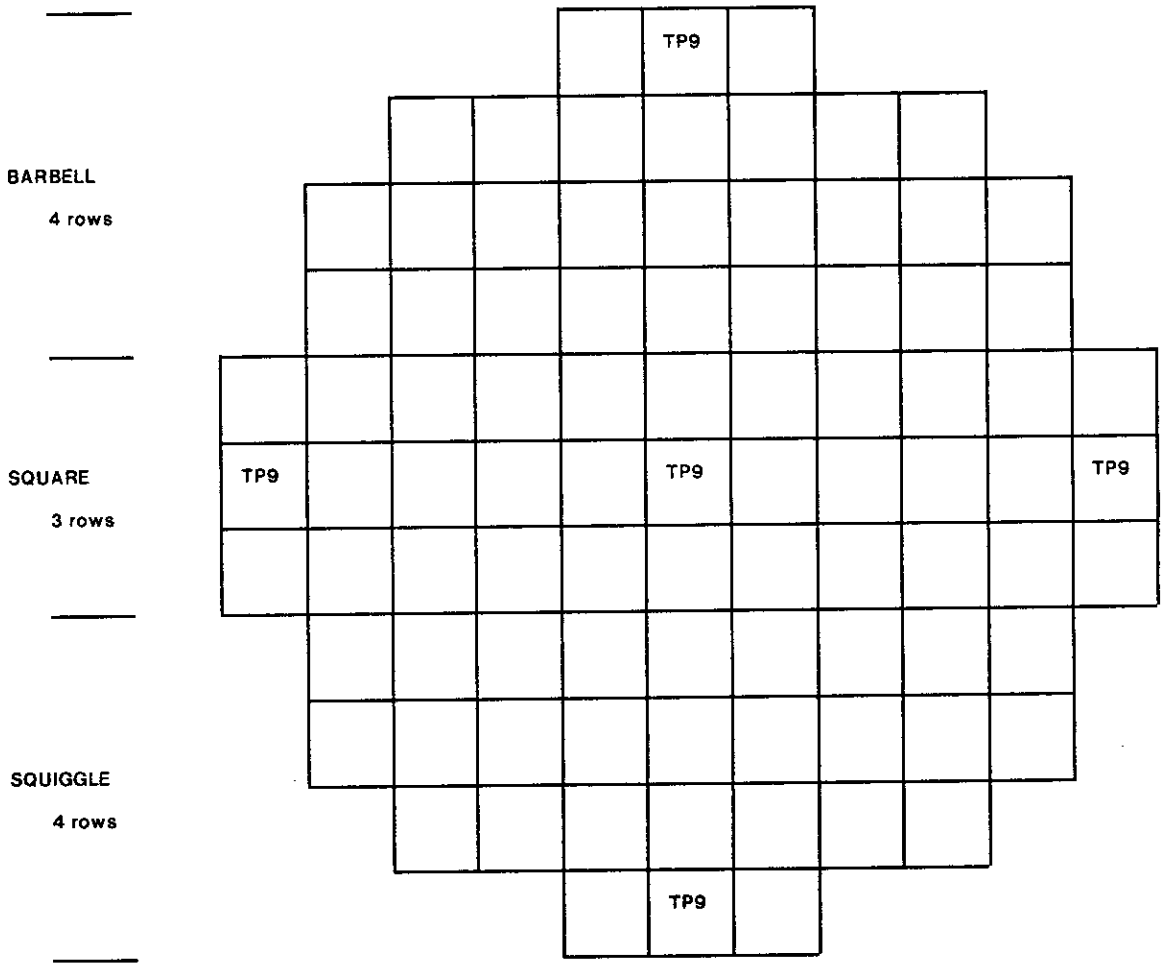
CD's are in this loction in all four corners and on all layers.

They are 6 micron bars with 6 micron gaps.



Pattern Identification Shape.

WAFER LAYOUT



Appendix B. Index of Manufacturers

MASKS

Micro Mask Inc.
695 Vaqueros Ave.
Sunnyvale, CA 94086
Phone: (408) 245-7342

Transmask Corp.
3952 Campus Drive
Newport Beach, CA 92660
Phone: (408) 247-4577 or (714) 540-6080

NBK Corp.
3010 Olcott
Santa Clara, CA 95051
Phone: (408) 988-2600

Ultratech Corp.
Photomask Division
1950 Coronado Drive
Santa Clara, CA 95051
Phone: (408) 247-0451

WAFER FAB

Maruman Integrated Circuits Inc.
1220 Midas Way
Sunnyvale, CA 94086
Phone: (408) 739-0560

Synertek Inc.
3001 Stender Way
Santa Clara, CA 95051
Phone: (408) 988-5600

Polycore Electronics Inc.
1107 Tourmaline Drive
Newbury Park, CA 91320
Phone: (213) 991-1061

VLSI Technology
10 Jackson Street, Suite 115
Los Gatos, CA 95030
Phone: (408) 395-5100

EQUIPMENT

ElectroMask
6109 De Soto Ave.
Woodland Hills, CA 91367
Phone: (213) 884-5050

GCA Corp.
174 Middlesex Turnpike
Burlington, MA 01803
Phone: (617) 272-5600

Perkin Elmer/ETEC
3392 Investment Blvd.
Hayward, CA 94545
Phone: (415) 783-9210

Varian/Extrion Division
Blackburn Industrial Park
Gloucester, MA 01930
Phone: (617) 281-2000

COMPUTING SERVICES

NCA
388 Oak Mead Parkway
Sunnyvale, CA 94086
Phone: (408) 245-7990

Appendix C. Mann 3000 Pattern Generator Format

The data format of the Mann 3000 Pattern Generator is a useful output format for maskmaking, since many mask makers have GCA Mann equipment, which can either work directly from this format or convert it appropriately. Electromask machines, which are also commonly used, have a similar but not identical format.

The following description is excerpted from "Type 3000 Pattern Generator Data Formats for Metric Units" from GCA/Burlington Division (there is also a format for English units, which we suggest you avoid). We assume use of 9-track tape, though the Mann 3000 can also accept paper tape or 7-track tape. The encoding is 800 bpi EBCDIC (not ASCII) with CRC and LRC characters, odd lateral parity and even longitudinal parity, and 512-character records (all this is standard data format for IBM-compatible magnetic tape).

Syntax

The true format is whatever the program accepts, which does not conform to a simple syntax, since it is not based on one; however, we have tried to formalize the format description, making safe assumptions where necessary. In the following syntax description, vertical bar | means *or*, curly brackets {} mean repetition zero or more times, nonterminals are written in lower case letters only, and everything else is a terminal.

```
tape        = <BOT> directory <EOF> {file <EOF>}
directory   = {entry} $
entry       = name filenumber numrecords
name        = namepart namepart
namepart    = cccccc | sccccc | ssccccc | sssccc | ssssscc | sssssc
filenumber  = digit digit digit digit
numrecords  = digit digit digit digit
file        = patternfile | otherfile
otherfile   = any legal EBCDIC characters
patternfile = { {newline} item {newline} } $
item        = message | exposure
message     = " {char} <CR>
exposure    = {parameter} ;
parameter   = X number | Y number | H number | W number | A number
newline     = <CR> | <LF>
number      = digit {digit}
char        = c | . | , | - | s
c           = letter | digit
s           = <SPACE>
letter      = A through Z
digit       = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Semantics

The *directory* identifies each *patternfile* with a unique *name* (such as "_CHIP2_MASK3") made of two 6-character *nameparts* (right justified and padded with spaces if shorter). The patternfiles are numbered in order of occurrence on the tape from 0001 (up to 2047), and the *filenumbers* are put in the directory with the names. The number of 512-character records in each patternfile is also in the directory; according to Mann documentation, *numrecords* is not used and need not be correct, except that it must be 4 decimal digits to fit the format. Each directory *entry* is exactly 20 characters, so if you stick to one design per *tape*, the directory will fit in one record; if you need 25 or more entries, fill the last 12 characters of the record with spaces and use another record, even if only for the terminal $ (yes, that really is a dollar sign to terminate the file; an EOF won't work as it did on older Mann machines).

A word of caution is in order; the GCA document does not say that directory entries should be in the same order as the files, and implies that *otherfiles* may be included on the tape but must not be mentioned in the directory. But the GCA document on the 3600 format, which is in many ways similar, says that otherfiles must have a corresponding 20-character directory entry (though the characters have no meaning), and implies that their entries are in order. To be safe, do not put files other than patternfiles on the tape, and put directory entries in order.

Each patternfile is a description of a pattern to be generated (a mask layer, or reticle, for IC applications). After the operator tells the machine which pattern to run, each *item* in the file causes the machine to take an action; *messages* cause typeout on the terminal, and *exposures* cause boxes to be flashed on the plate (glass photographic plate). Messages are not of much use, except to type limericks to the operator (notice that spaces are allowed only in messages, not in exposures or between items). Exposures are the meat of the pattern generator format, and their semantics depend on the coordinate system.

The coordinate system is left-handed, with X increasing to the left and Y increasing upward as you look at the plate. All dimensions are positive, so the origin is the lower right corner. The center coordinate, for any plate size, is half the maximum coordinate. Thus the origin is not generally anywhere within a pattern, though it will be exactly at the corner of a maximum size pattern (10 cm square). See figure C.1 for the relation of the axes, the plate, and a typical exposure.

The units of measure for X and Y, which represent the center of a box, are 1 micron at the reticle (0.1 micron at the chip for 10X reticles); the range is 0 to 100000, which allows features to actually extend outside the 10 cm field, as long as their centers are within the field. All *numbers* are decimal integers, most significant *digit* first. Leading zeros should be suppressed, since the format actually restricts the number of digits that may be used to six after X or Y, four after H or W, and three after A (the digit-count limit was not indicated in the above syntax, is not a sufficient restriction, and may not even be necessary).

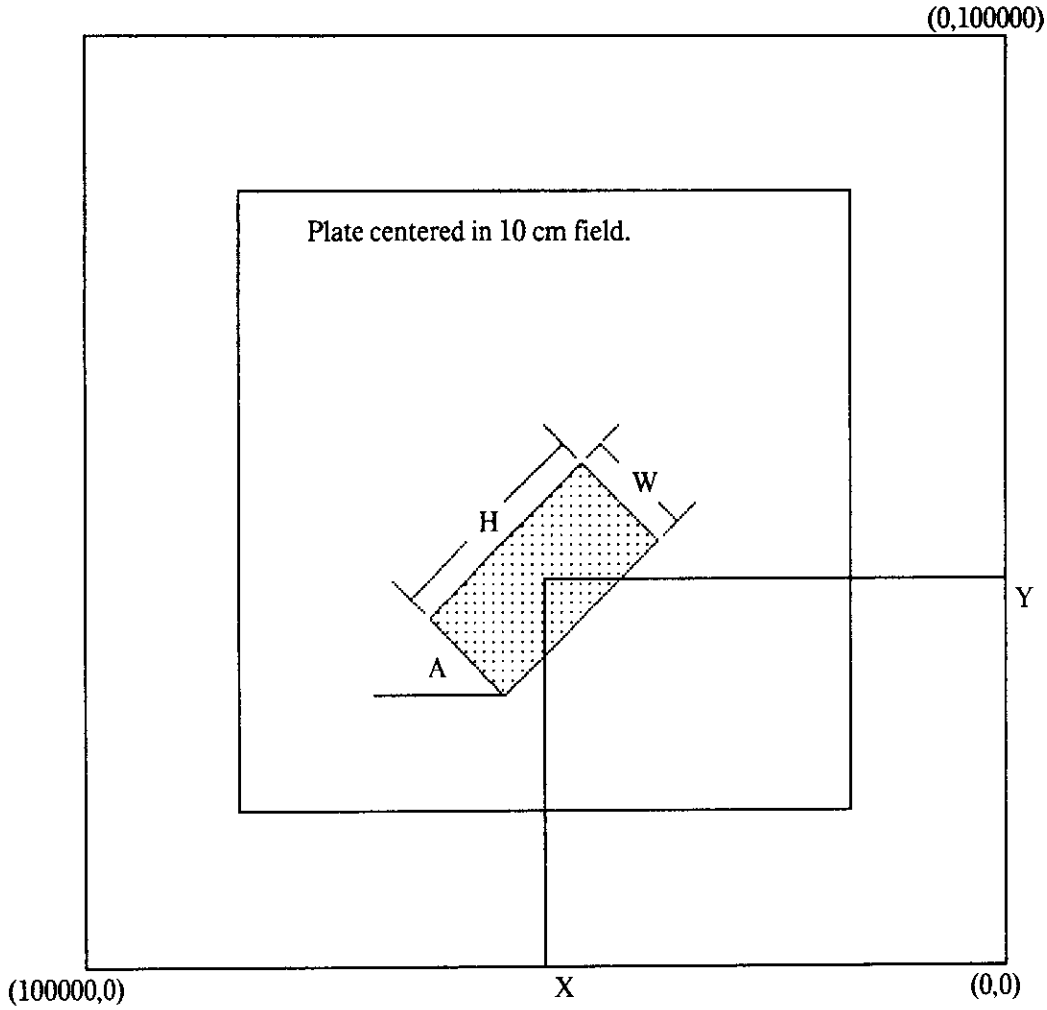


Figure C.1
Mann 3000 Coordinate System.

The height H and width W are measured in the same units as X and Y, but the range is 4 microns to 3000 microns. W is the box size along the X axis before rotation, and H is the perpendicular dimension.

The angle of rotation A is in units of 0.1 degree clockwise, in the range of 0 to 899 (89.9 degrees). If CIF is converted using dimensions directly and angles ArcTan(dir.X,dir.Y), which are counterclockwise from the +X axis, the result is the proper pattern, but mirrored. If X coordinates are then negated to fix mirroring, angles should also be negated (and everything converted to proper ranges).

As the syntax shows, each exposure consists of any number of *parameters*, terminated by a semicolon. The machine keeps track of the latest value of each of the five parameters X, Y, H, W, and A, and generates the aperature accordingly (the first exposure must specify all five). Thus, if no parameters are listed between semicolons, the same aperture should be flashed over again; in fact, the pattern generator notices this and prints a "format exception" error message. If some parameter is listed more than once between semicolons, only the latest value is used (this was useful for correcting mistakes when punching paper tapes).

CIF to Mann conversion

The conversion of CIF files to Mann format should soon be a widely available utility. Hopefully, this format description will help make it happen sooner. A major stumbling block in this effort will be *sorting*, since the pattern generator works very slowly if the exposures are not well sorted relative to the way the machine likes to work. Unfortunately, GCA has not been willing to provide an algorithm for doing a good sort. Due to the way the machine works, it is easy to sort wrong, i.e. much worse than no sort at all.

We do know some basic strategies for sorting, which we summarize here. The X dimension is called the *scan direction*, and the Y dimension is called the *stepover direction*, implying that the machine likes to scan continuously in X, while flashing everything within a narrow band in Y. But changing aperture size or angle is slow, and if a change is not finished by the time the X coordinate is reached an *overrun* occurs. Since the machine is moving in the X direction, overruns cannot be flashed immediately; rather than stop and come back, the machine saves the exposure in a buffer. When the overrun buffer is full, those exposures must then be done (which can be very slow, since they may be anywhere). To accommodate this feature, sort primarily on angle, then on aperture size, so changes occur only rarely (this also reduces the number of parameters in the patternfile). Then within each aperture sort spatially by whatever algorithm you think might work; don't try to outguess the machine.

Appendix D. A Basic Library of Symbol Layouts

[This information was first distributed as an informational message to participants of the MPC79 project, a multiuniversity multiproject chip set involving Stanford, CMU, MIT, Berkeley, U. of Rochester, and other schools. It is the official release announcement and documentation of the file of library symbol designs provided for use in that project. Both black-and-white and color checkplots have been added to this document to clarify the descriptions of the layouts. The black-and-white plots contain only outlines of symbols and boxes to help in interpreting the color plots.]

Summary

A single file in CIF 2.0 format is provided for your use. It contains standard I/O Pads, all the pieces needed to make PLA's, shift register cells on a pitch compatible with the PLA, and superbuffers for driving clock and control lines. The intention is that these should be a sufficient set of common cell designs to allow implementation of combinational functions and state machines simply by placement and interconnection, thus allowing students to focus their efforts on the architecture, logic, and cell designs specific to their own projects. They also serve as examples, and can be used to test your CIF plotting software.

Project lab coordinators at each participating school should retrieve the file from [MAXC]<MPC79>LIBRARY79-250.CIF (250 is the value of lambda in CIF units, which is 2.5 microns). Additional hardcopy documentation with color checkplots will be mailed later to each school.

Conventions

Since the library symbols were designed using ICARUS, they all have names in addition to numbers. Names are represented in the CIF file by the use of a userExtensionCommand, in the format "9 name;". These names may be used or ignored, but in this message all symbols are referred to by name. The terms symbol and cell are used interchangeably in this message.

In all cases, the origin of a symbol is the upper left corner of its minimum bounding box; hence, all Y coordinates in the CIF library symbol definitions are negative.

Since this library is intended to be compatible with even the simplest design systems, no geometric primitives other than boxes with default direction and no rotation transformations except multiples of 90 degrees are used; all box edges before and after transformation lie on the lambda grid.

Plots of various symbols should be made from the CIF file to serve as the illustrations for this document.

Pad Descriptions

Bonding pads and associated circuitry are provided for input, output, clocked output, tristate input/output, Vdd, ground, and conversion of a single-phase clock input to two-phase. A standard configuration was chosen to simplify placement and interconnection of the pads. See PadBlank, which is called by most of the other pads, as an example:

```
DS 2;  9 PadBlank;
( 4 items. ); (bounding box 0, 0 to 26500, -26500);
L NM; B L 26500 W 2000 C 13250,-1000; (Vdd line);
L NM; B L 20500 W 2000 C 13250,-25500; (ground line);
L NM; B L 13500 W 13500 C 13250,-13250; (metal pad);
L NG; B L 11500 W 11500 C 13250,-13250; (overglass window);
DF;
```

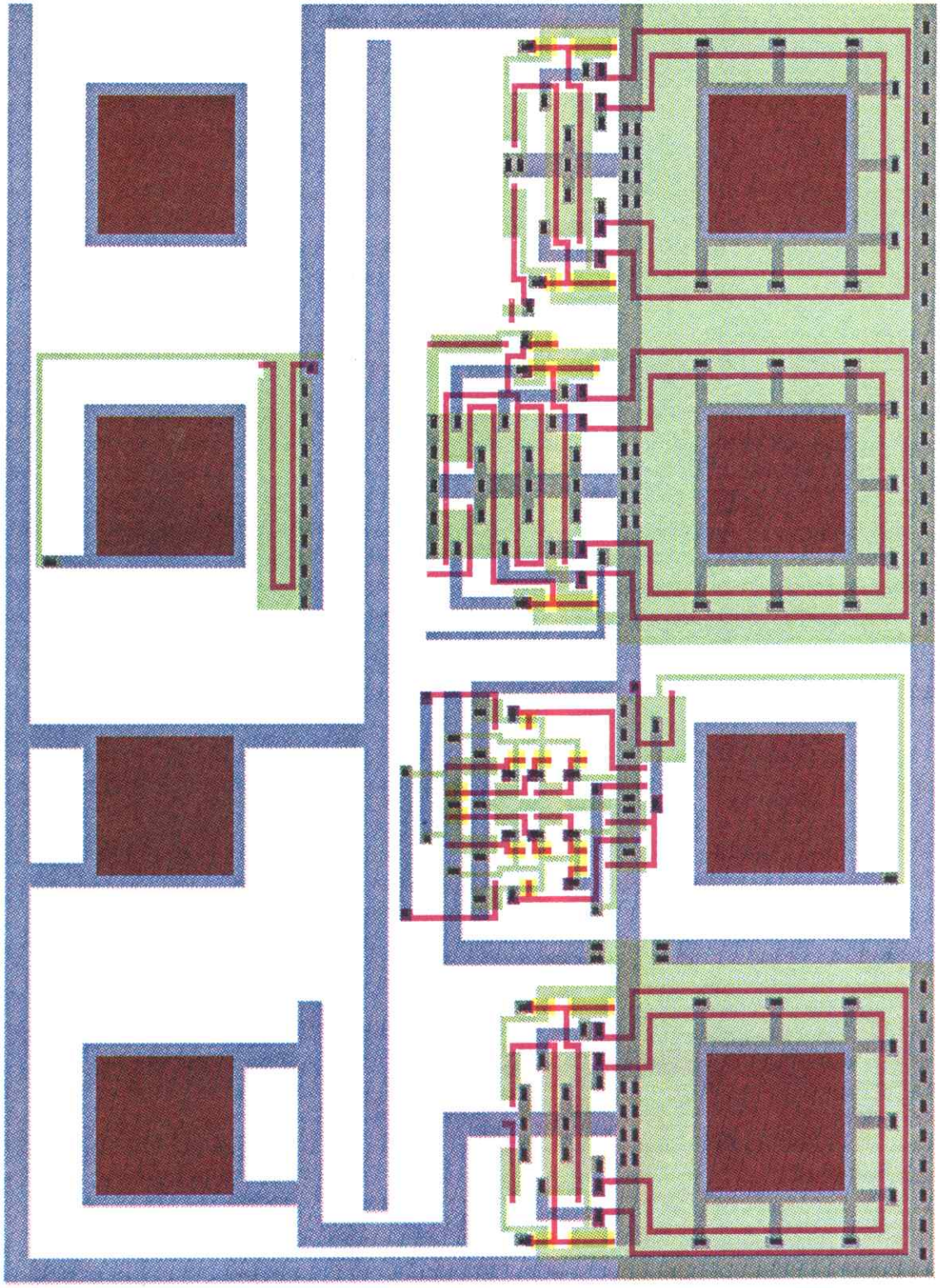
PadBlank illustrates the fact that each pad is a 135 micron metal square with a 115 micron square overglassing window in a 265 micron (106 lambda) square area, with horizontal metal lines along the top and bottom edges. The top metal line, which is always used for a Vdd connection, crosses the entire width of the symbol, and defines the outside edge of the project of which the symbol is a part. The default orientation is correct for pads along the top edge of a project. The bottom metal line is used for ground, and stops short of the edges of the symbol to facilitate running Vdd around the corners of a project without going outside the bounding box of the pad symbols. A typical project will have abutting pads around two, three, or four sides, with an 8-lambda Vdd ring around the outside, and an 8-lambda ground ring around the inside; pads should only be placed around the perimeter of a project, since interior pads are difficult to bond. The Vdd pad omits the ground line so that there will be a gap in the ground ring to bring power into the project. See PadSample for an example of all the pads and their power connections.

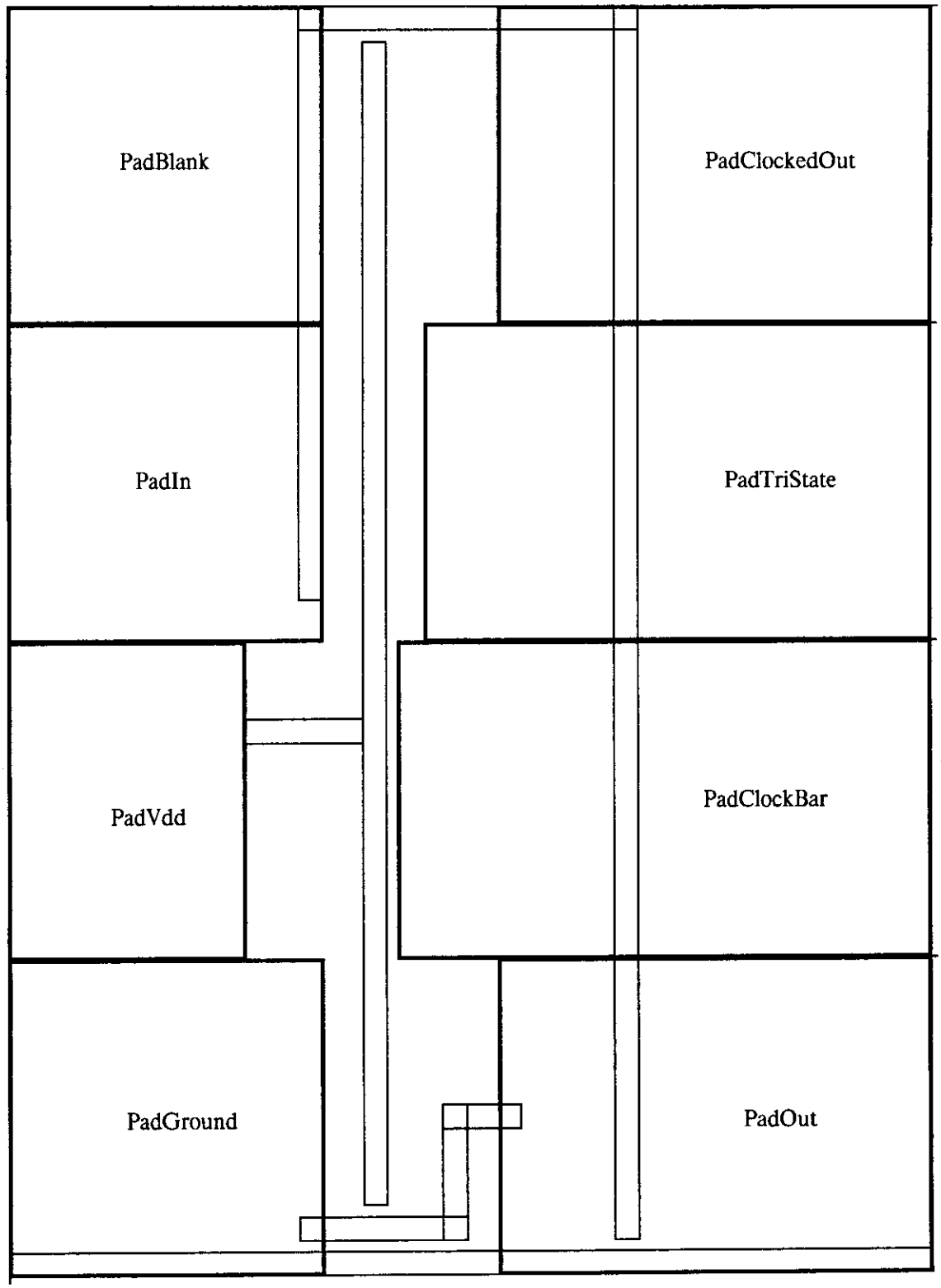
The pads and their sizes (in lambda) are as follows:

PadBlank	106x106
PadGround	106x106
PadVdd	106x80
PadIn	106x106
PadDriver	106x106
PadOut	106x145
PadClockedOut	106x145
PadTriState	106x170
PadClockBar	106x179

The output pads call PadDriver, which uses enhancement-mode pullups, so the output levels are TTL-like; internally these pads should be driven from level-restored signals. The input pad does no level restoration (it simply provides a lightning arrestor), so inputs from TTL-like devices should connect only to $k=8$ logic, and should not control pass transistors.

Note that PadClockBar generates inverse clocks, guaranteed to never both be low at the same time, from a single-phase TTL-compatible input; these are driven by a powerful superbuffer for distribution around a chip, and are intended to be used with InvertingSB (described below) to generate clocks and gated control signals. Designers should carefully consider the implications of using this clock generation circuit before including it in their projects; it results in considerably less clocking flexibility than using separate input pads for the clock phases.




PadBlank
PadClockedOut
PadIn
PadTriState
PadVdd
PadClockBar
PadGround
PadOut

PadSample

PLA Descriptions

The PLA symbols provided for MPC79 were designed to be simple and clean, and are not as small as they could be in some cases. The pitch of the metal and poly lines in both planes is 8 lambda, when 7 lambda would be possible. This extra spacing makes layout of the edge cells on the same pitch much easier, and makes possible the layout of a shift register cell on the same pitch as the PLA inputs (16 lambda). The overall structure and orientation of the PLA is similar to that shown in Mead&Conway's *Introduction to VLSI Systems*, pp. 102-107 (inputs and outputs on the bottom edge, AND-plane on the left, OR-plane on the right); but, as can be seen by comparing the layouts, the extra spacing simplifies most of the cells.

See PLA-4-8-8 (a 4-input, 8-product term, 8-output PLA) as an example of how the pieces fit together. This layout illustrates the use of extra metal ground meshing that may be needed in large PLA's; typically a ground line for every 32 product term lines will be adequate, but a conservative designer might use more frequent ground lines. This layout also illustrates all the possible clocked and unclocked input and output cells, and the NOR output cells (which, if used in place of the usual inverters, effectively AND pairs of adjacent OR-plane outputs to facilitate "folding" of ROMs). For simplicity, no provision is made for an odd number of lines across either plane in either direction.

The basic cells provided are the following (cells marked with * should be rotated 90 degrees clockwise for use in the OR-plane):

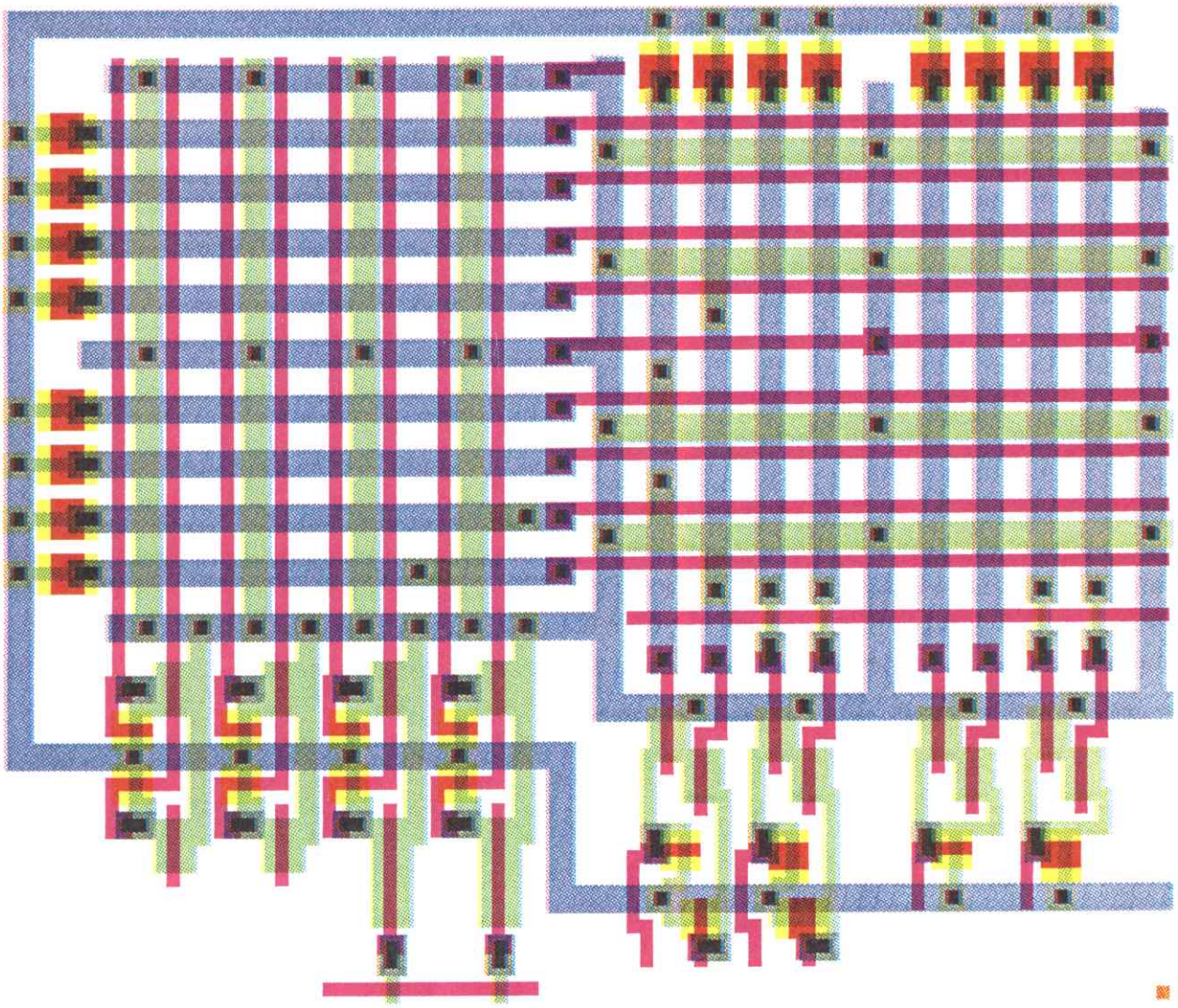
PlaCell*
PlaGround*
PlaPullups*
PlaConnect
PlaIn
PlaClockedIn
PlaOut
PlaClockedOut
PlaNorOut
PlaClockedNorOut
PlaHoleWires

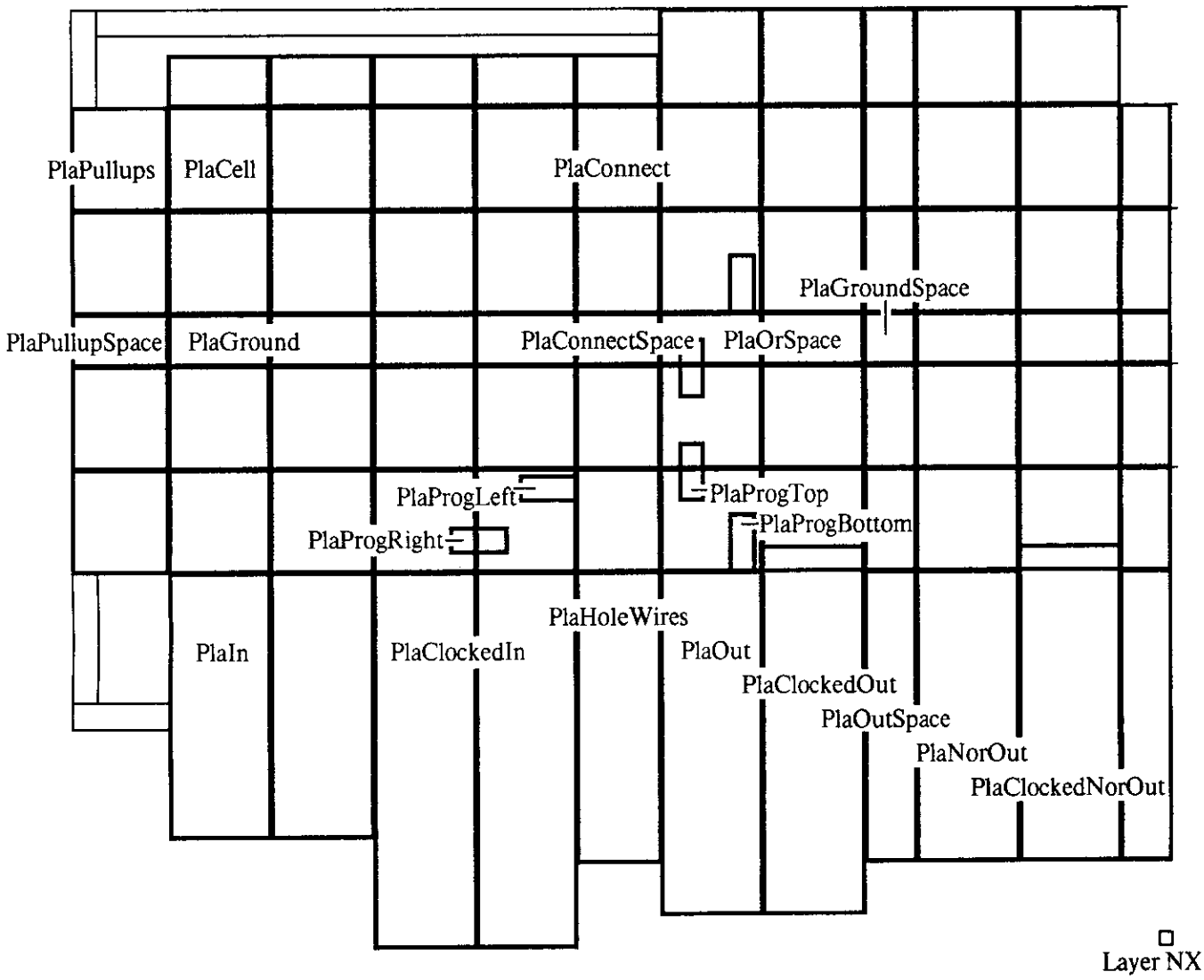
These are the programming cells for the left and right sides of the AND-plane cells and the top and bottom of the OR-plane cells:

PlaProgLeft
PlaProgRight
PlaProgTop
PlaProgBottom

In addition, cells are provided to fill the spaces left to accomodate the optional extra ground meshing:

PlaOrSpace
PlaConnectSpace
PlaGroundSpace*
PlaPullupSpace*
PlaOutSpace




PlaPullups
PlaCell
PlaConnect
PlaGroundSpace
PlaPullupSpace
PlaGround
PlaConnectSpace
PlaOrSpace
PlaProgLeft
PlaProgTop
PlaProgBottom
PlaProgRight
PlaHoleWires
PlaIn
PlaClockedIn
PlaOut
PlaClockedOut
PlaOutSpace
PlaNorOut
PlaClockedNorOut
Layer NX

Pla-4-8-8

Superbuffer Descriptions

A set of superbuffers is provided for use as clock and control line drivers. They were optimized for flexibility and regularity, rather than absolute speed. SuperBuffer is a subcell of both InvertingSB and NoninvertingSB, and has no output structure of its own. With alternate cells mirrored, superbuffers fit together with their diffusion outputs regularly spaced 16 lambda apart along the top edge. To simplify placement, mirrored pairs are provided for both inverting and noninverting types.

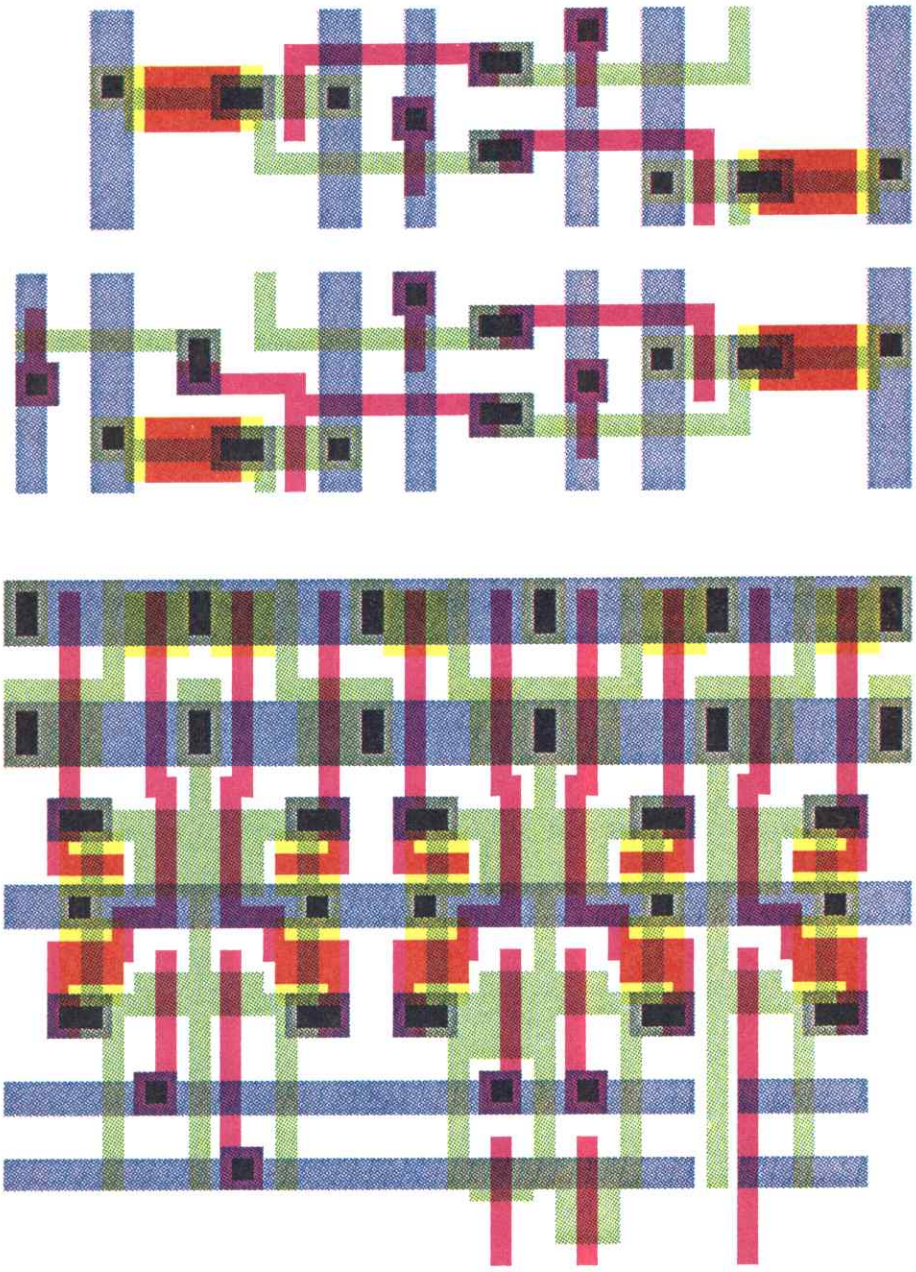
The symbol SBExample illustrates the use of superbuffers with various input options. Generally, it is intended that Phi1inverse and Phi2inverse from PadClockBar would be distributed on the metal lines that (partially) cross the bottom edge of the superbuffers, and that clock gating signals would be routed in poly from below (low-true logic) into InvertingSB, making it into a NOR driver. Thus both gated and nongated clocks are driven through the same circuit, with similar delay (but use caution in loading these signals, since fatal clock skew is still possible).

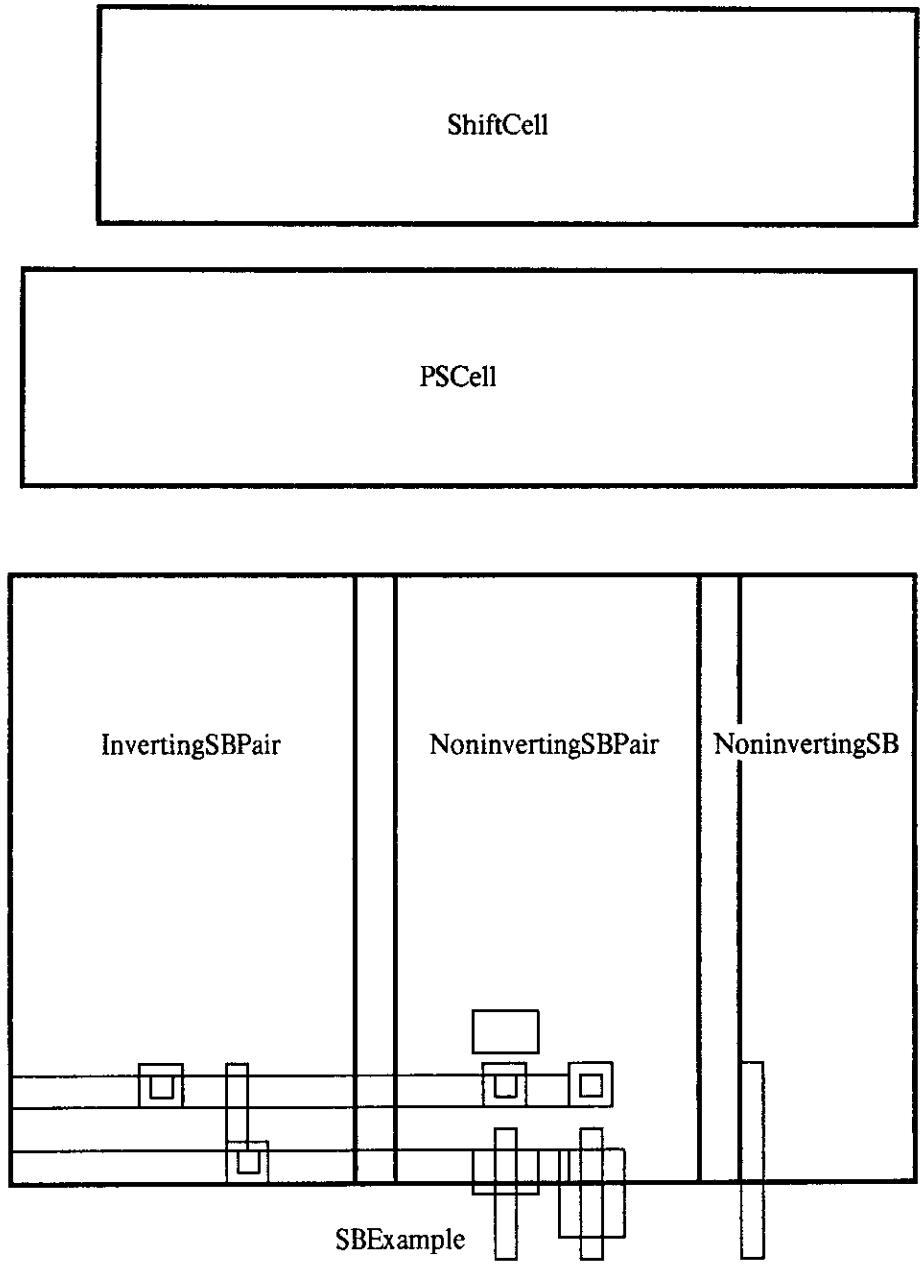
The symbol names are as follows:
SuperBuffer
InvertingSB
InvertingSBPair
NoninvertingSB
NoninvertingSBPair
SBExample

Shift Register Descriptions

These full-bit shift register cells fit together with a pitch of 16 lambda, which make them useful for serial-to-parallel conversion (ShiftCell) on the pitch of PlaIn or PlaClockedIn, or for parallel-to-serial conversion (PSCell) on the pitch of PlaNorOut or PlaClockedNorOut. These cells have vertical metal power and clock lines, parallel to the direction of shift; they should be rotated to interface with the PLA in standard orientation.

The shift register symbol names are as follows:
ShiftCell
PSCell




ShiftCell
PSCell
InvertingSBPair
NoninvertingSBPair
NoninvertingSB
SBExample

Checking

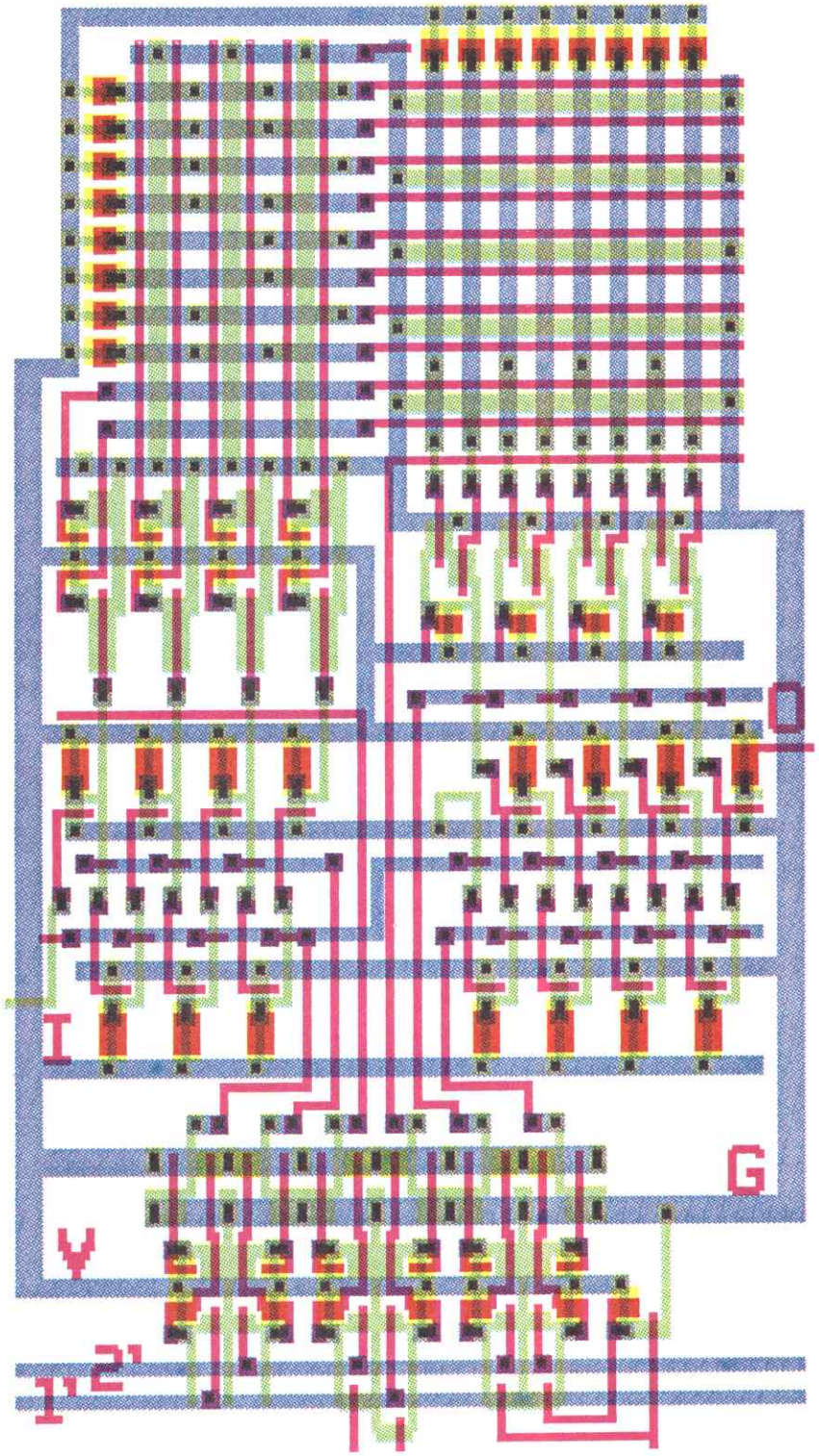
It is recommended that each designer take the time to examine library cells in detail before using them, to avoid misuse and to look for possible errors or incompatibilities. Although we have checked the cells carefully, they are not guaranteed to be free of logic, circuit, or design rule errors; if any errors are found, please notify MPC79@PARC immediately (the first person to report each fatal error will be amply rewarded). If anyone documents the cells in more detail, such as coordinates of connection points, etc., we would be glad to collect and distribute that information.

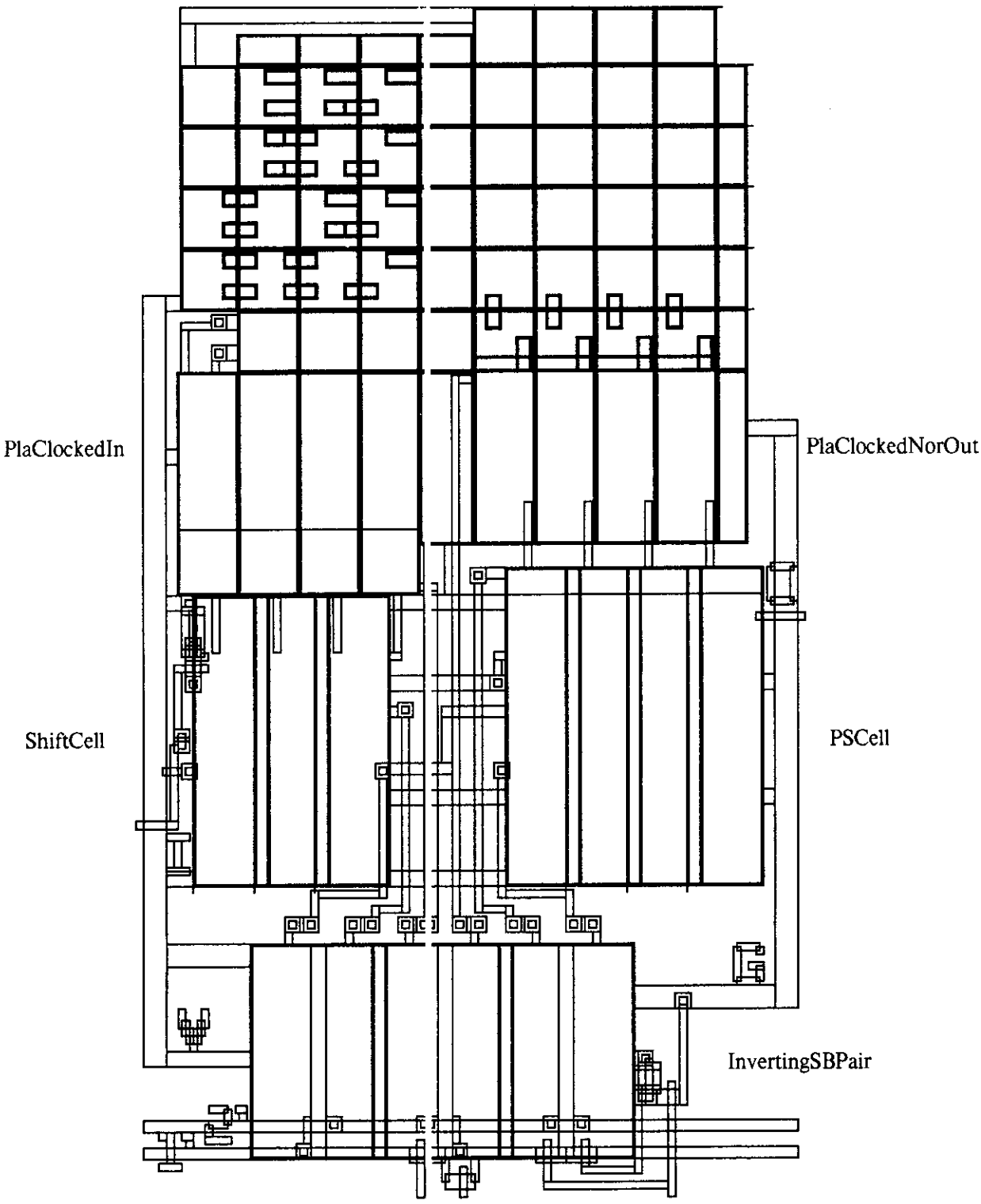
It would be useful if instructors would assign homework problems based on the library, such as to look for errors, to analyze the output pad or the clock pad, to develop formulas for the size and speed of the PLA, to analyze capacitive coupling of the clock to the storage node in ShiftCell, or to look at current limits of the power lines in various cells. We would be glad to see the results of such assignments.

Example

The last pair of figures shows a small project (named "Demo") assembled from library cells, to better illustrate how they can be interconnected to perform a non-trivial function. The example design implements a "serial ROM", which accepts a serial address input and delivers a serial data output (with 4 bits per word in this case). It illustrates the use of NOR outputs on the PLA to allow an 8x8 PLA (ROM) to implement a 16x4 ROM, and the use of both serial-to-parallel and parallel-to-serial conversion forms of the shift register, with clock gating and buffering included. This project is incomplete, as it still needs pads and programming of the ROM data.

The CIF text of the library file follows, with the example project included at the end of the file.


D
I
G
V
I'2'


PlaClockedIn
PlaClockedNorOut
ShiftCell
PSCell
InvertingSBPair

```
( CIF 2.0 );
( Filed on [MAXC]<MPC79>LIBRARY79-250.CIF );
( Icarus file [MAXC]<MPC79>LIBRARY79-3.IC );
( Created by Sif from library79-5.ic );
( FAB nMOS, lambda=2.5 microns );
( modified by Lyon:  October 12, 1979  1:08 PM );
( more comments added October 22, 1979  4:09 PM );
( userExtensionCommand 9 is Icarus symbol name );
( Squished  January 24, 1980  11:11 AM );

DS 1;
9 ClockLogic;
L NM;  B 10000 1500 5000,-14750;
       B 10000 1500 5000,-17000;
       B 9750 2000 4875,-2250;
L NP;  B 500 4500 250,-15500;
       B 3000 500 1750,-13500;
L ND;  B 2000 2750 1750,-14125;
L NM;  B 1500 1000 2000,-12000;
L NP;  B 750 1000 1625,-12000;
       B 5250 500 4125,-3750;
       B 500 1250 1750,-11125;
L NC;  B 500 500 1750,-12000;
       B 500 1000 1750,-14750;
L NP;  B 500 7250 1750,-7125;
L NI;  B 1500 1500 2500,-10750;
L NP;  B 1500 500 2500,-10750;
L ND;  B 1000 1000 2250,-12000;
L NC;  B 500 500 2250,-12000;
L ND;  B 500 2500 2500,-10500;
       B 500 750 2500,-12625;
       B 1750 500 3375,-9500;
       B 1000 1500 4000,-17000;
       B 500 4250 3750,-14875;
L NC;  B 500 1000 4000,-17000;
L ND;  B 500 7000 4250,-9750;
       B 2500 500 5500,-11750;
       B 2500 500 5500,-9500;
       B 2500 500 5500,-6500;
       B 1000 1500 5250,-14750;
       B 4750 2000 7375,-2250;
L NI;  B 1500 1500 5750,-11750;
       B 1500 1500 5750,-9500;
       B 1500 1500 5750,-6500;
L NC;  B 500 1000 5250,-14750;
       B 500 1000 5500,-2250;
L ND;  B 1500 4000 6250,-15250;
L NP;  B 1250 500 6125,-12500;
       B 1250 500 6125,-10250;
       B 1250 500 6125,-7250;
       B 500 1500 5750,-11750;
       B 500 1500 5750,-9500;
       B 500 1500 5750,-6500;
       B 500 5000 6250,-15250;
       B 500 3000 6500,-2250;
L NM;  B 1000 1500 7000,-12250;
L NP;  B 1000 750 7000,-12625;
L NM;  B 1000 1500 7000,-10000;
L NP;  B 1000 750 7000,-10375;
L ND;  B 1000 1000 7000,-9750;
       B 1000 1000 7000,-12000;
L NM;  B 1000 1500 7000,-7000;
L NP;  B 1000 750 7000,-7375;
L ND;  B 1000 1000 7000,-6750;
L NC;  B 500 500 7000,-12000;
       B 500 500 7000,-12500;
       B 500 500 7000,-9750;
       B 500 500 7000,-10250;
L ND;  B 500 1250 7000,-17125;
L NC;  B 500 500 7000,-6750;
       B 500 500 7000,-7250;
L ND;  B 2250 1000 8125,-16750;
L NP;  B 1750 500 7875,-11000;
L ND;  B 1000 500 7500,-9500;
       B 500 4500 7250,-4500;
L NP;  B 1500 500 8000,-7250;
L ND;  B 750 500 7625,-11750;
L NM;  B 2250 1000 8875,-5000;
L ND;  B 1500 2000 8500,-12500;
       B 1500 2000 8500,-8750;
L NI;  B 1500 2000 8500,-16750;
L NP;  B 1000 1000 8250,-5000;
       B 500 4000 8000,-2750;
L NC;  B 500 500 8250,-5000;
L NP;  B 500 6750 8500,-14375;
       B 500 2750 8500,-8875;
L NC;  B 500 1000 9000,-2250;
L NM;  B 10000 1500 14000,-14750;
       B 10000 1500 14000,-17000;
       B 9750 1000 13875,-5000;
       B 9750 2000 13875,-2250;
L ND;  B 1000 1500 9500,-17000;
       B 1000 14250 9500,-8375;
L NC;  B 500 1000 9500,-14750;
       B 500 1000 9500,-17000;
L ND;  B 2500 1000 10500,-1250;
       B 2250 1000 10875,-16750;
       B 1500 2000 10500,-12500;
       B 1500 2000 10500,-8750;
L NI;  B 1500 2000 10500,-16750;
L NC;  B 500 1000 10000,-2250;
L ND;  B 4000 2000 12000,-2750;
L NP;  B 1750 500 11125,-11000;
       B 1500 500 11000,-7250;
       B 500 6750 10500,-14375;
       B 500 2750 10500,-8875;
       B 4000 500 12750,-250;
       B 500 4250 11000,-2125;
L ND;  B 1000 500 11500,-9500;
       B 750 500 11375,-11750;
L NM;  B 1000 1500 12000,-12250;
L NP;  B 1000 750 12000,-12625;
L NM;  B 1000 1500 12000,-10000;
L NP;  B 1000 750 12000,-10375;
L ND;  B 1000 1000 12000,-9750;
       B 1000 1000 12000,-12000;
L NM;  B 1000 1500 12000,-7000;
L NP;  B 1000 750 12000,-7375;
L ND;  B 1000 1000 12000,-6750;
       B 500 4500 11750,-4500;
L NC;  B 500 500 12000,-12000;
       B 500 500 12000,-12500;
       B 500 500 12000,-9750;
       B 500 500 12000,-10250;
L ND;  B 500 1250 12000,-17125;
L NC;  B 500 500 12000,-6750;
       B 500 500 12000,-7250;
L ND;  B 1500 4000 12750,-15250;
L NP;  B 5250 500 14875,-5000;
L ND;  B 2500 500 13500,-11750;
       B 2500 500 13500,-9500;
       B 1750 500 13125,-6500;
L NP;  B 1250 500 12875,-12500;
       B 1250 500 12875,-10250;
       B 1250 500 12875,-7250;
       B 500 3750 12500,-3125;
L NI;  B 1500 1500 13250,-11750;
       B 1500 1500 13250,-9500;
       B 1500 1500 13250,-6500;
L NP;  B 500 5000 12750,-15250;
L ND;  B 2250 3000 14125,-2250;
L NP;  B 500 1500 13250,-11750;
       B 500 1500 13250,-9500;
       B 500 1500 13250,-6500;
L ND;  B 1000 1500 13750,-14750;
L NC;  B 500 1000 13500,-2250;
       B 500 1000 13750,-14750;
L ND;  B 500 3000 14000,-8000;
       B 1750 500 14875,-6500;
L NI;  B 1500 1500 14750,-6500;
L NP;  B 500 4250 14500,-2125;
       B 1250 500 15125,-7250;
L ND;  B 1000 1500 15000,-17000;
       B 500 3750 14750,-11375;
L NP;  B 500 1500 14750,-6500;
L ND;  B 1750 500 15625,-9500;
```

```
L NC;  B 500 1000 15000,-17000;
L ND;  B 3250 500 16625,-3500;
       B 500 4250 15250,-14875;
L NM;  B 1000 1500 16000,-7000;
L NP;  B 1000 750 16000,-7375;
L ND;  B 1000 1000 16000,-6750;
L NM;  B 1000 1500 16000,-5750;
L NP;  B 3000 500 17250,-13500;
L NI;  B 1500 1500 16500,-10750;
L NP;  B 1500 500 16500,-10750;
L NC;  B 500 500 16000,-6750;
       B 500 500 16000,-7250;
L ND;  B 2000 2750 17250,-14125;
L NM;  B 1500 1000 17000,-12000;
L ND;  B 1000 1000 16750,-12000;
       B 500 2500 16500,-10500;
       B 500 750 16500,-12625;
L NC;  B 500 500 16750,-12000;
L NP;  B 750 1000 17375,-12000;
       B 500 1250 17250,-11125;
L NC;  B 500 500 17250,-12000;
       B 500 1000 17250,-14750;
L NP;  B 500 5750 17250,-7875;
L ND;  B 1000 1000 18250,-5000;
       B 500 1250 18000,-4125;
L NC;  B 500 500 18250,-5000;
L NP;  B 500 4500 18750,-15500;
DF;


DS 2;
9 PadBlank;
L NM;  B 26500 2000 13250,-1000;
       B 20500 2000 13250,-25500;
       B 13500 13500 13250,-13250;
L NG;  B 11500 11500 13250,-13250;
DF;


DS 3;
9 PadDriver;
C 2;
L ND;  B 26500 7000 13250,-3500;
       B 7000 19500 3500,-16750;
L NC;  B 1000 500 2000,-1000;
L NP;  B 500 23000 2250,-14000;
       B 4250 500 4125,-25250;
       B 22500 500 13250,-2500;
L NM;  B 1000 1500 3250,-7000;
       B 1000 1500 3250,-13250;
       B 1000 1500 3250,-19500;
L NC;  B 500 1000 3250,-7000;
       B 500 1000 3250,-13250;
       B 500 1000 3250,-19500;
L NM;  B 3250 1000 4875,-7000;
       B 3250 1000 4875,-13250;
       B 3250 1000 4875,-19500;
L NC;  B 1000 500 4500,-1000;
L NP;  B 500 19500 4250,-14250;
       B 4750 500 6375,-23750;
       B 18500 500 13250,-4500;
       B 500 1500 6000,-25750;
L NM;  B 1500 1000 7000,-3500;
       B 1000 2750 7000,-5125;
L NC;  B 1000 500 7000,-3500;
       B 1000 500 7000,-1000;
L ND;  B 12500 7000 13250,-23000;
L NP;  B 500 3000 8500,-25000;
L NC;  B 1000 500 9500,-1000;
       B 1000 500 10250,-26000;
       B 1000 500 10250,-25000;
       B 1000 500 12000,-1000;
       B 1000 500 12250,-26000;
       B 1000 500 12250,-25000;
L NM;  B 1500 1000 13250,-3500;
L NC;  B 1000 500 13250,-3500;
L NM;  B 1000 2750 13250,-5125;
L NC;  B 1000 500 14250,-26000;
       B 1000 500 14250,-25000;
       B 1000 500 14500,-1000;
       B 1000 500 16250,-26000;
       B 1000 500 16250,-25000;
       B 1000 500 17000,-1000;
L NP;  B 4750 500 20125,-23750;
       B 500 3000 18000,-25000;
L NM;  B 1500 1000 19500,-3500;
L NC;  B 1000 500 19500,-3500;
L NM;  B 1000 2750 19500,-5125;
L NC;  B 1000 500 19500,-1000;
L ND;  B 7000 19500 23000,-16750;
L NM;  B 3250 1000 21625,-19500;
       B 3250 1000 21625,-13250;
       B 3250 1000 21625,-7000;
L NP;  B 4250 500 22375,-25250;
       B 500 1500 20500,-25750;
L NC;  B 1000 500 22000,-1000;
L NP;  B 500 19500 22250,-14250;
L NM;  B 1000 1500 23250,-7000;
       B 1000 1500 23250,-13250;
       B 1000 1500 23250,-19500;
L NC;  B 500 1000 23250,-7000;
       B 500 1000 23250,-13250;
       B 500 1000 23250,-19500;
       B 1000 500 24500,-1000;
L NP;  B 500 23000 24250,-14000;
DF;


DS 4;
9 PadOut;
C 3;
L ND;  B 1000 6750 2250,-29875;
       B 2500 1000 3500,-32750;
       B 2000 3000 3500,-28750;
L NI;  B 1500 4000 3500,-28750;
       B 1500 2000 3500,-32750;
L NM;  B 1000 1500 3500,-34250;
L NP;  B 1000 750 3500,-33875;
L ND;  B 1500 1000 3750,-34500;
L NP;  B 500 7000 3500,-30250;
L NC;  B 500 1000 3500,-34250;
L NP;  B 3500 500 5250,-30750;
L ND;  B 500 3750 4500,-34375;
       B 2000 1000 5250,-29250;
       B 3000 500 6000,-36000;
L NM;  B 1500 2250 5500,-28625;
L NP;  B 1500 1000 5500,-28000;
L NC;  B 1000 500 5500,-28000;
       B 1000 500 5500,-29250;
L NM;  B 1000 3750 5750,-30625;
       B 3500 1000 7000,-32750;
L NP;  B 500 2250 6000,-27125;
       B 500 3500 6750,-33500;
       B 16500 500 14750,-31750;
       B 5250 500 9125,-35250;
       B 13250 500 13125,-29750;
       B 500 1000 6750,-30250;
L ND;  B 6000 4000 10250,-30750;
       B 4000 1750 9250,-35375;
       B 1500 1000 8000,-32750;
L NM;  B 3250 1000 8875,-28000;
L NP;  B 1500 1000 8000,-28000;
L NC;  B 1000 500 8000,-32750;
       B 1000 500 8000,-28000;
L NP;  B 500 2250 8500,-27125;
L ND;  B 1500 1250 9750,-28125;
L NC;  B 1000 500 9750,-28000;
L NM;  B 6500 1000 13250,-30750;
       B 6500 1000 13250,-34250;
L ND;  B 6500 1000 13250,-34250;
L NC;  B 1000 500 10750,-30750;
       B 1000 500 10750,-34250;
L NM;  B 2000 8500 13250,-30250;
L NC;  B 1000 500 13250,-30750;
       B 1000 500 13250,-34250;
L NP;  B 500 1250 13250,-35625;
```

```
L ND;  B 6000 4000 16250,-30750;
L NP;  B 6500 500 16500,-35250;
L NC;  B 1000 500 15750,-30750;
L ND;  B 4000 1750 17250,-35375;
L NC;  B 1000 500 15750,-34250;
L ND;  B 1500 1250 16750,-28125;
L NM;  B 3250 1000 17625,-28000;
L NC;  B 1000 500 16750,-28000;
L NP;  B 500 2250 18000,-27125;
L ND;  B 1500 1000 18500,-32750;
L NM;  B 3500 1000 19500,-32750;
L NP;  B 1500 1000 18500,-28000;
L NC;  B 1000 500 18500,-32750;
       B 1000 500 18500,-28000;
L ND;  B 3000 500 20500,-29000;
       B 3000 500 20500,-36000;
L NP;  B 500 2250 20500,-27125;
L NM;  B 1000 5000 20750,-30000;
L NP;  B 1500 1000 21000,-28000;
L NM;  B 1500 1000 21000,-28000;
L NC;  B 1000 500 21000,-28000;
L ND;  B 2500 1000 23000,-32750;
       B 500 3750 22000,-34375;
       B 2000 3000 23000,-28750;
       B 1500 1000 22750,-34500;
L NI;  B 1500 4000 23000,-28750;
       B 1500 2000 23000,-32750;
L NM;  B 1000 1500 23000,-34250;
L NP;  B 1000 750 23000,-33875;
       B 500 7000 23000,-30250;
L NC;  B 500 1000 23000,-34250;
L ND;  B 1000 6750 24250,-29875;
DF;


DS 5;
9 PadIn;
C 2;
L ND;  B 19250 4500 12625,-23250;
L NC;  B 1000 500 3750,-25000;
L NP;  B 500 1750 5000,-23125;
       B 18750 500 14125,-22250;
       B 18750 500 14125,-24000;
L NC;  B 1000 500 6250,-25000;
L NM;  B 1000 4250 7000,-5125;
L ND;  B 1000 2000 7000,-3500;
L NC;  B 500 1000 7000,-3750;
L ND;  B 16750 500 15625,-2750;
L NC;  B 1000 500 8750,-25000;
       B 1000 500 11250,-25000;
       B 1000 500 13750,-25000;
       B 1000 500 16250,-25000;
       B 1000 500 18750,-25000;
       B 1000 500 21250,-25000;
L ND;  B 2000 3250 23000,-23125;
L NP;  B 1000 1000 23000,-25500;
L NC;  B 500 500 23000,-25500;
L NP;  B 500 1250 23250,-21625;
       B 500 2000 23250,-25000;
L ND;  B 500 24000 24000,-14500;
DF;


DS 6;
9 PadGround;
C 2;
L NM;  B 2000 4500 7500,-22250;
       B 2000 4500 19000,-22250;
DF;


DS 7;
9 PadVdd;
L NM;  B 26500 2000 13250,-1000;
       B 13500 13500 13250,-13250;
       B 2000 4500 7500,-4250;
L NG;  B 11500 11500 13250,-13250;
L NM;  B 2000 4500 19000,-4250;
DF;


DS 8;
9 PadTriState;
C 3;
L ND;  B 500 4750 750,-34375;
L NM;  B 1500 1000 1250,-34250;
L ND;  B 1000 1000 1000,-34250;
L NI;  B 1500 2500 1500,-32750;
L NC;  B 1000 500 1250,-34250;
L ND;  B 1500 6000 1500,-39250;
       B 1000 1500 1500,-32750;
L NP;  B 500 2500 1500,-32750;
       B 750 1000 1625,-34250;
       B 500 6750 1500,-39125;
       B 750 500 2125,-34500;
L ND;  B 1000 7000 2500,-30000;
       B 17250 1000 10625,-42000;
L NP;  B 500 1750 2500,-35125;
       B 3500 500 4000,-35750;
L ND;  B 2000 1000 3500,-33000;
       B 2000 3000 3500,-30000;
L NI;  B 1500 2000 3500,-33000;
       B 1500 4000 3500,-30000;
L NP;  B 500 6250 3500,-31125;
L NM;  B 1000 6750 3750,-37125;
L NP;  B 1000 750 3750,-34125;
L ND;  B 1500 1000 4000,-34750;
L NP;  B 3250 500 5125,-32000;
L NC;  B 500 1000 3750,-34500;
L ND;  B 2000 1000 5250,-31000;
L NM;  B 4500 1000 6500,-40000;
L ND;  B 500 2750 4750,-33875;
L NP;  B 1500 1000 5500,-29500;
L NM;  B 1500 2500 5500,-30250;
L NC;  B 1000 500 5500,-29500;
       B 1000 500 5500,-31000;
L NM;  B 1000 6500 5750,-33250;
L NP;  B 500 6250 5750,-37875;
       B 14250 500 12625,-41000;
       B 14250 500 12875,-35000;
       B 500 2750 6000,-27875;
L NM;  B 2750 1000 7375,-36000;
L NP;  B 500 2000 6750,-38000;
       B 14750 500 13875,-37000;
       B 500 2250 6750,-33875;
       B 5250 500 9125,-39000;
       B 500 1500 6750,-31500;
       B 13000 500 13250,-31000;
       B 13000 500 13250,-33000;
L NM;  B 1500 3500 8000,-30750;
L ND;  B 4000 4250 9250,-40375;
       B 1750 7000 8125,-35000;
L NP;  B 1500 1000 8000,-29500;
L NM;  B 12000 1000 13250,-42000;
L NC;  B 1000 500 8000,-32000;
       B 1000 500 8000,-36000;
       B 1000 500 8000,-40000;
       B 1000 500 8000,-42000;
       B 1000 500 8000,-29500;
L NP;  B 500 2750 8500,-27875;
L ND;  B 8500 9000 13250,-34000;
L NM;  B 6500 1000 13250,-30000;
       B 6500 1000 13250,-34000;
       B 6500 1000 13250,-38000;
L NC;  B 1000 500 10500,-42000;
       B 1000 500 10750,-34000;
       B 1000 500 10750,-30000;
       B 1000 500 10750,-38000;
L NM;  B 2000 15500 13250,-34250;
L NC;  B 1000 500 13250,-42000;
       B 1000 500 13250,-34000;
       B 1000 500 13250,-30000;
       B 1000 500 13250,-38000;
L NP;  B 6000 500 17750,-39000;
L NC;  B 1000 500 15750,-34000;
       B 1000 500 15750,-30000;
```

```
        B 1000 500 15750,-38000;
L ND;   B 4000 4250 17250,-40375;
L NC;   B 1000 500 16000,-42000;
L ND;   B 1750 8250 18375,-34375;
L NM;   B 1500 3500 18500,-30750;
        B 3500 1000 19500,-36000;
        B 4750 1000 20125,-40000;
L NP;   B 500 2750 18000,-27875;
        B 1500 1000 18500,-29500;
L NC;   B 1000 500 18500,-32000;
        B 1000 500 18500,-36000;
        B 1000 500 18500,-40000;
        B 1000 500 18500,-42000;
        B 1000 500 18500,-29500;
L ND;   B 1500 1750 19250,-27375;
L NM;   B 1500 1000 19250,-27750;
L NC;   B 1000 500 19250,-27750;
L ND;   B 1500 1000 19750,-32000;
L NP;   B 500 2250 19750,-33875;
L NM;   B 5750 750 22625,-27875;
L ND;   B 750 1750 20375,-31625;
        B 2250 1000 21125,-31000;
L NM;   B 1000 6750 20750,-33125;
L NP;   B 500 2750 20500,-27875;
        B 1500 1000 21000,-29500;
L NM;   B 1500 1000 21000,-29500;
        B 1500 1000 21000,-29500;
L NP;   B 500 3750 20500,-40625;
L NC;   B 1000 500 21000,-29500;
L NP;   B 500 5500 21250,-34500;
        B 2000 500 22000,-32000;
L ND;   B 500 2750 22000,-33875;
        B 1500 1000 22750,-34750;
        B 2000 1000 23000,-33000;
        B 2000 3000 23000,-30000;
L NI;   B 1500 2000 23000,-33000;
        B 1500 4000 23000,-30000;
L NM;   B 1000 6750 23000,-37125;
L NP;   B 1000 750 23000,-34125;
L NC;   B 500 1000 23000,-34500;
L NP;   B 500 6250 23000,-31125;
L ND;   B 1000 7000 24250,-30000;
L NM;   B 750 15000 25625,-35000;
DF;


DS 9;
9 PadClockBar;
C 2;
L ND;   B 500 20750 2750,-12875;
        B 1500 500 3500,-23000;
L NP;   B 1000 1000 3500,-25000;
L ND;   B 17000 500 11500,-2750;
L NM;   B 1000 12250 3500,-32625;
L NC;   B 500 500 3500,-25000;
L NP;   B 5250 500 5875,-24750;
L NM;   B 12750 1000 10125,-42500;
L NP;   B 1000 1000 4250,-42500;
        B 500 1000 4000,-41500;
        B 4500 500 6000,-22000;
C 1 T 3750,-23250;
L NC;   B 500 500 4250,-42500;
L ND;   B 5500 5750 7000,-23625;
L NC;   B 1000 500 5500,-26000;
L NM;   B 8000 1000 10000,-23250;
L NC;   B 1000 500 6750,-23250;
        B 1000 500 7500,-26000;
L NP;   B 500 3250 8250,-23375;
L ND;   B 1000 1000 10500,-44250;
L NM;   B 12750 1000 16375,-44250;
L NC;   B 500 500 10500,-44250;
L ND;   B 500 2750 10750,-42375;
L NP;   B 1500 1000 13250,-23250;
L NC;   B 1000 500 13250,-23250;
L NP;   B 500 500 14250,-23500;
L ND;   B 1000 1000 16000,-42500;
        B 500 1000 15750,-41500;
L NC;   B 500 500 16000,-42500;
L NM;   B 1000 3500 19500,-4750;
L ND;   B 1000 1500 19500,-3750;
L NC;   B 500 1000 19500,-3750;
L NP;   B 1000 1000 22250,-44250;
L NC;   B 500 500 22250,-44250;
L NP;   B 500 2750 22500,-42375;
L NM;   B 1750 1500 23625,-40250;
        B 2000 21500 25500,-12750;
        B 2000 13500 25500,-34250;
L ND;   B 2000 7000 25500,-25500;
L NC;   B 500 1000 25000,-22750;
        B 500 1000 25000,-28250;
        B 500 1000 26000,-22750;
        B 500 1000 26000,-28250;
DF;


DS 10;
9 PadClockedOut;
C 3;
L ND;   B 1000 6750 2250,-29875;
        B 2500 1000 3500,-32750;
        B 2000 3000 3500,-28750;
L NI;   B 1500 4000 3500,-28750;
        B 1500 2000 3500,-32750;
L NM;   B 1000 1500 3500,-34250;
L NP;   B 1000 750 3500,-33875;
L ND;   B 1500 1000 3750,-34500;
L NP;   B 500 7000 3500,-30250;
L NC;   B 500 1000 3500,-34250;
L NP;   B 3500 500 5250,-30750;
L ND;   B 500 3750 4500,-34375;
        B 2000 1000 5250,-29250;
        B 3000 500 6000,-36000;
L NM;   B 1500 2250 5500,-28625;
L NP;   B 1500 1000 5500,-28000;
L NC;   B 1000 500 5500,-28000;
        B 1000 500 5500,-29250;
L NM;   B 1000 3750 5750,-30625;
        B 3500 1000 7000,-32750;
L NP;   B 500 2250 6000,-27125;
        B 500 3500 6750,-33500;
        B 13250 500 13125,-31750;
        B 5250 500 9125,-35250;
        B 13250 500 13125,-29750;
        B 500 1000 6750,-30250;
L ND;   B 6000 4000 10250,-30750;
        B 4000 1750 9250,-35375;
        B 1500 1000 8000,-32750;
L NM;   B 3250 1000 8875,-28000;
L NP;   B 1500 1000 8000,-28000;
L ND;   B 5250 500 9875,-34500;
L NC;   B 1000 500 8000,-32750;
        B 1000 500 8000,-28000;
L NP;   B 500 2250 8500,-27125;
L ND;   B 1500 1250 9750,-28125;
L NC;   B 1000 500 9750,-28000;
L NM;   B 6500 1000 13250,-30750;
L NC;   B 1000 500 10750,-30750;
L NM;   B 2000 10250 13250,-31125;
L ND;   B 2000 2000 13250,-35250;
L NC;   B 1000 500 13250,-30750;
        B 1000 500 13250,-34750;
        B 1000 500 13250,-35750;
L ND;   B 6000 4000 16250,-30750;
        B 5250 500 16625,-36000;
L NP;   B 8250 500 18875,-35250;
L NC;   B 1000 500 15750,-30750;
L ND;   B 4000 1750 17250,-35125;
        B 1500 1250 16750,-28125;
L NM;   B 3250 1000 17625,-28000;
L NC;   B 1000 500 16750,-28000;
L NP;   B 500 2250 18000,-27125;
L ND;   B 1500 1000 18500,-32750;
L NM;   B 3500 1000 19500,-32750;
L NP;   B 1500 1000 18500,-28000;
L NC;   B 1000 500 18500,-32750;
        B 1000 500 18500,-28000;
```

```
L ND;  B 3000 500 20500,-29000;
       B 3000 500 20500,-34500;
L NP;  B 500 1000 19750,-31500;
       B 3500 500 21250,-30750;
       B 500 2250 20500,-27125;
L NM;  B 1000 5000 20750,-30000;
L NP;  B 1500 1000 21000,-28000;
L NM;  B 1500 1000 21000,-28000;
L NC;  B 1000 500 21000,-28000;
L ND;  B 500 3000 22000,-33250;
       B 1750 1000 22625,-33500;
       B 2500 500 23000,-32000;
       B 2000 3000 23000,-28750;
L NI;  B 1500 4000 23000,-28750;
       B 1500 1500 23000,-32000;
L NP;  B 2250 500 23625,-34500;
       B 500 750 22750,-34875;
L NM;  B 1000 1500 23000,-33250;
L NP;  B 1000 750 23000,-32875;
       B 500 6000 23000,-29750;
L NC;  B 500 1000 23000,-33250;
L ND;  B 1000 5750 24250,-29375;
L NM;  B 1500 1000 25000,-34000;
L NP;  B 750 1000 24625,-34000;
       B 2000 500 25250,-35500;
L NC;  B 1000 500 25000,-34000;
L ND;  B 1000 2750 25250,-34875;
DF;

DS 11;
9 PadSample;
C 6 R 0,1 T 0,-106000;
C 7 R 0,1 T 0,-79500;
C 5 R 0,1 T 0,-53000;
C 2 R 0,1 T 0,-26500;
L NM;  B 77750 2000 38875,-105000;
       B 10250 2000 24875,-60500;
       B 2000 49500 25500,-24750;
       B 28750 2000 38875,-1000;
       B 14250 2000 31625,-102000;
       B 2000 97000 31000,-51500;
C 9 R 0,-1 T 77750,-53000;
C 8 R 0,-1 T 77750,-26500;
       B 2000 11250 37750,-97375;
       B 6500 2000 40000,-92750;
C 4 R 0,-1 T 77750,-79500;
C 10 R 0,-1 T 77750,0;
       B 2000 103000 52250,-51500;
DF;

DS 12;
9 PlaCell;
L NM;  B 4000 1000 2000,-750;
       B 4000 1000 2000,-2750;
L NP;  B 500 4000 500,-2000;
L ND;  B 1000 4000 1500,-2000;
L NP;  B 500 4000 2500,-2000;
DF;

DS 13;
9 PlaGround;
L NM;  B 4000 1000 2000,-750;
L NP;  B 500 2000 500,-1000;
L ND;  B 1000 2000 1500,-1000;
L NC;  B 500 500 1500,-750;
L NP;  B 500 2000 2500,-1000;
DF;

DS 14;
9 PlaPullups;
L NM;  B 1000 4000 500,-2000;
L ND;  B 1000 1000 500,-750;
       B 1000 1000 500,-2750;
L NC;  B 500 500 500,-2750;
       B 500 500 500,-750;
L ND;  B 2250 500 1875,-750;
       B 2250 500 1875,-2750;
L NI;  B 2250 1500 2375,-750;
       B 2250 1500 2375,-2750;
L NP;  B 1250 1500 2375,-750;
       B 1250 1500 2375,-2750;
L NM;  B 1500 1000 3000,-750;
       B 1500 1000 3000,-2750;
L NC;  B 1000 500 3000,-2750;
       B 1000 500 3000,-750;
L ND;  B 1000 1000 3250,-2750;
       B 1000 1000 3250,-750;
DF;

DS 15;
9 PlaConnect;
L NM;  B 1250 1000 625,-750;
       B 1250 1000 625,-2750;
L NP;  B 1000 1000 750,-750;
       B 1000 1000 750,-2750;
L NC;  B 500 500 750,-750;
       B 500 500 750,-2750;
L NP;  B 2250 500 2125,-500;
       B 2250 500 2125,-2500;
L NM;  B 1000 4000 2500,-2000;
L ND;  B 1250 1000 2625,-1500;
L NC;  B 500 500 2500,-1500;
DF;

DS 16;
9 PlaConnectSpace;
L NM;  B 2250 1000 1125,-750;
L NP;  B 1000 1000 750,-750;
L NC;  B 500 500 750,-750;
L NP;  B 2250 500 2125,-500;
L NM;  B 1000 2000 2500,-1000;
DF;

DS 17;
9 PlaOrSpace;
L NP;  B 4000 500 2000,-500;
L NM;  B 1000 2000 1250,-1000;
       B 1000 2000 3250,-1000;
DF;

DS 18;
9 PlaGroundSpace;
L NP;  B 2000 500 1000,-500;
L NM;  B 1000 2000 1250,-1000;
L NP;  B 1000 1000 1250,-500;
L NC;  B 500 500 1250,-500;
DF;

DS 19;
9 PlaIn;
L NM;  B 4000 1000 2000,-750;
       B 4000 1000 2000,-5500;
L NP;  B 1000 1000 500,-8000;
       B 1000 1000 500,-3000;
       B 500 1500 250,-7000;
       B 500 1500 250,-4000;
       B 2250 500 1375,-6500;
L NM;  B 1500 1000 1000,-8000;
L NI;  B 1500 1500 1000,-6500;
       B 1500 1500 1000,-4500;
L NP;  B 1500 500 1000,-4500;
L NM;  B 1500 1000 1000,-3000;
L NP;  B 500 2750 500,-1375;
L NC;  B 1000 500 1000,-8000;
L ND;  B 1000 1000 1000,-5500;
L NC;  B 1000 500 1000,-3000;
L ND;  B 1000 1500 1250,-7750;
```

```
        B 1000 1500 1250,-3250;
        B 500 1500 1000,-6500;
        B 500 250 1000,-7125;
L NC;   B 500 500 1000,-5500;
L ND;   B 500 1500 1000,-4750;
        B 500 250 1000,-3875;
        B 1000 1250 1500,-625;
L NC;   B 500 500 1500,-750;
L ND;   B 1500 2000 2500,-3000;
        B 1500 2000 2500,-8750;
L NP;   B 500 6750 2500,-3375;
        B 500 3000 2500,-8750;
L ND;   B 1000 1000 3500,-750;
        B 750 7750 3375,-4875;
L NC;   B 500 500 3500,-750;
DF;


DS 20;
9 PlaClockedIn;
L NP;   B 4000 500 2000,-14000;
C 19;
L ND;   B 1500 2000 2500,-10750;
L NM;   B 1000 1500 2500,-12750;
L ND;   B 1000 1000 2500,-13000;
L NP;   B 1000 750 2500,-12375;
        B 500 2500 2500,-11000;
L NC;   B 500 1000 2500,-12750;
L ND;   B 500 1250 2500,-13875;
DF;


DS 21;
9 PlaProgTop;
L ND;   B 1000 2250 500,-1125;
L NC;   B 500 500 500,-500;
DF;


DS 22;
9 PlaProgBottom;
L ND;   B 1000 2250 500,-1125;
L NC;   B 500 500 500,-1750;
DF;


DS 23;
9 PlaProgLeft;
L ND;   B 2250 1000 1125,-500;
L NC;   B 500 500 500,-500;
DF;


DS 24;
9 PlaProgRight;
L ND;   B 2250 1000 1125,-500;
L NC;   B 500 500 1750,-500;
DF;


DS 25;
9 PlaPullupSpace;
L NM;   B 1000 2000 500,-1000;
        B 1000 1000 3250,-750;
DF;


DS 26;
9 PlaOut;
L NM;   B 4000 1000 2000,-3750;
L NP;   B 4000 500 2000,-500;
        B 500 3000 250,-10500;
L NM;   B 4000 1000 2000,-10750;
L ND;   B 500 3250 750,-6875;
L NM;   B 1000 1500 1000,-8750;
L ND;   B 1000 1000 1000,-8500;
L NP;   B 1000 750 1000,-9125;
        B 500 1750 750,-12375;
L ND;   B 1500 2000 1500,-4750;
L NM;   B 1000 2500 1250,-1250;
L NP;   B 1000 1000 1250,-2000;
L ND;   B 1500 500 1500,-8250;
L NC;   B 500 1000 1000,-8750;
L ND;   B 1000 1000 1250,-10750;
L NC;   B 500 500 1250,-2000;
        B 500 500 1250,-10750;
L NP;   B 500 4000 1500,-4250;
        B 1500 500 2000,-9000;
L NI;   B 1500 1500 2000,-9000;
L ND;   B 750 500 1625,-10250;
        B 2250 500 2375,-11500;
        B 500 2250 2000,-9375;
        B 1000 1000 2500,-3750;
        B 1500 2000 2750,-6250;
L NI;   B 1500 1500 2750,-11500;
L NC;   B 500 500 2500,-3750;
L NM;   B 1500 1000 3000,-12500;
L NP;   B 750 1000 2625,-12500;
        B 500 3250 2750,-6125;
L NC;   B 1000 500 3000,-12500;
L NP;   B 500 500 2750,-13000;
        B 500 1500 2750,-11500;
        B 750 500 3125,-4750;
L NM;   B 1000 1000 3250,-2000;
L NP;   B 1000 1000 3250,-2000;
L NM;   B 1000 2500 3250,-1250;
L ND;   B 1000 1000 3250,-12500;
L NC;   B 500 500 3250,-2000;
L ND;   B 500 5500 3500,-9500;
L NP;   B 500 2750 3500,-3625;
DF;


DS 27;
9 PlaClockedOut;
L NM;   B 4000 1000 2000,-4750;
L NP;   B 4000 500 2000,-1500;
        B 500 3000 250,-11500;
L NM;   B 4000 1000 2000,-11750;
L ND;   B 500 3250 750,-7875;
L NM;   B 1000 1500 1000,-9750;
L ND;   B 1000 1000 1000,-9500;
L NP;   B 1000 750 1000,-10125;
        B 500 1750 750,-13375;
L ND;   B 1500 2000 1500,-5750;
        B 1000 1000 1250,-500;
L NM;   B 1000 1000 1250,-500;
L ND;   B 1000 1000 1250,-2500;
L NP;   B 1000 750 1250,-3125;
L NM;   B 1000 1500 1250,-2750;
L ND;   B 1500 500 1500,-9250;
L NC;   B 500 1000 1000,-9750;
L ND;   B 1000 1000 1250,-11750;
L NC;   B 500 500 1250,-500;
L ND;   B 500 1500 1250,-1500;
L NC;   B 500 1000 1250,-2750;
        B 500 500 1250,-11750;
L NP;   B 500 4000 1500,-5250;
        B 1500 1000 2000,-10250;
L NI;   B 1500 2000 2000,-10250;
L ND;   B 750 500 1625,-11250;
        B 2250 500 2375,-12500;
L NI;   B 2000 1500 2500,-12500;
L ND;   B 500 2250 2000,-10375;
        B 1000 1000 2500,-4750;
        B 1500 2000 2750,-7250;
L NP;   B 1000 1500 2500,-12500;
L NC;   B 500 500 2500,-4750;
L NM;   B 1500 1000 3000,-13500;
L NP;   B 750 1000 2625,-13500;
        B 500 3250 2750,-7125;
L NC;   B 1000 500 3000,-13500;
L NP;   B 500 500 2750,-14000;
        B 750 500 3125,-5750;
L ND;   B 1000 1000 3250,-500;
L NM;   B 1000 1000 3250,-500;
```

```
L ND;  B 1000 1000 3250,-2600;
L NP;  B 1000 750 3250,-3125;
L NM;  B 1000 1500 3250,-2750;
L ND;  B 1000 1000 3250,-13500;
L NC;  B 500 500 3250,-500;
L ND;  B 500 1500 3250,-1500;
L NC;  B 500 1000 3250,-2750;
L ND;  B 500 5500 3500,-10500;
L NP;  B 500 2750 3500,-4625;
DF;


DS 28;
9 PlaHoleWires;
L NM;  B 1250 1000 625,-5500;
       B 2250 1000 1125,-750;
       B 1000 5250 750,-8625;
       B 2000 1000 2250,-10750;
       B 1000 3500 2500,-1750;
       B 1250 1000 2625,-3750;
DF;


DS 29;
9 PlaOutSpace;
L NP;  B 2000 500 1000,-500;
L NM;  B 2000 1000 1000,-3750;
       B 2000 1000 1000,-10750;
       B 1000 3500 1250,-1750;
DF;


DS 30;
9 PlaNorOut;
L NM;  B 4000 1000 2000,-3750;
L NP;  B 4000 500 2000,-500;
L NM;  B 4000 1000 2000,-10750;
L ND;  B 500 3000 750,-6750;
L NM;  B 1000 1500 1000,-8750;
L ND;  B 1000 1000 1000,-8500;
L NP;  B 1000 750 1000,-9125;
       B 500 2250 750,-10125;
L ND;  B 1500 2000 1500,-4750;
L NM;  B 1000 2500 1250,-1250;
L NP;  B 1000 1000 1250,-2000;
L NC;  B 500 1000 1000,-8750;
       B 500 500 1250,-2000;
L ND;  B 2500 500 2250,-8250;
L NP;  B 500 4000 1500,-4250;
       B 1500 500 2000,-9000;
L NI;  B 1500 1500 2000,-9000;
L ND;  B 1000 1000 2000,-10750;
       B 500 2750 2000,-9625;
L NC;  B 500 500 2000,-10750;
L ND;  B 1000 1000 2500,-3750;
       B 1500 2000 2750,-6250;
L NC;  B 500 500 2500,-3750;
L NP;  B 500 3250 2750,-6125;
       B 750 500 3125,-4750;
L NM;  B 1000 1000 3250,-2000;
L NP;  B 1000 1000 3250,-2000;
L NM;  B 1000 2500 3250,-1250;
L NC;  B 500 500 3250,-2000;
L ND;  B 500 1750 3500,-7625;
L NP;  B 500 2750 3500,-3625;
DF;


DS 31;
9 PlaClockedNorOut;
L NP;  B 4000 500 2000,-1500;
L NM;  B 4000 1000 2000,-4750;
       B 4000 1000 2000,-11750;
L ND;  B 500 3000 750,-7750;
L NM;  B 1000 1500 1000,-9750;
L ND;  B 1000 1000 1000,-9500;
L NP;  B 1000 750 1000,-10125;
       B 500 2250 750,-11125;
L ND;  B 1000 1000 1250,-500;
L NM;  B 1000 1000 1250,-500;
L ND;  B 1000 1000 1250,-2500;
L NP;  B 1000 750 1250,-3125;
L NM;  B 1000 1500 1250,-2750;
L ND;  B 1500 2000 1500,-5750;
L NC;  B 500 1000 1000,-9750;
       B 500 500 1250,-500;
L ND;  B 500 1500 1250,-1500;
L NC;  B 500 1000 1250,-2750;
L ND;  B 2500 500 2250,-9250;
L NP;  B 500 4000 1500,-5250;
       B 1500 1000 2000,-10250;
L NI;  B 1500 2000 2000,-10250;
L ND;  B 1000 1000 2000,-11750;
       B 500 2750 2000,-10625;
L NC;  B 500 500 2000,-11750;
L ND;  B 1500 2000 2750,-7250;
       B 1000 1000 2500,-4750;
L NC;  B 500 500 2500,-4750;
L NP;  B 500 3250 2750,-7125;
L ND;  B 1000 1000 3250,-500;
L NM;  B 1000 1000 3250,-500;
L ND;  B 1000 1000 3250,-2500;
L NP;  B 1000 750 3250,-3125;
L NM;  B 1000 1500 3250,-2750;
L NP;  B 750 500 3125,-5750;
L NC;  B 500 500 3250,-500;
L ND;  B 500 1500 3250,-1500;
L NC;  B 500 1000 3250,-2750;
L NP;  B 500 2750 3500,-4625;
L ND;  B 500 1750 3500,-8625;
DF;


DS 32;
9 Pla-4-8-8;
L NM;  B 3750 1000 1875,-27250;
C 25 T 0,-11750;
C 14 T 0,-13750;
C 14 T 0,-17750;
C 14 T 0,-3750;
C 14 T 0,-7750;
       B 1000 3750 500,-1875;
       B 1000 5000 500,-24250;
       B 22000 1000 12000,-500;
C 12 T 3750,-3750;
C 12 T 3750,-7750;
C 12 T 3750,-13750;
C 12 T 3750,-17750;
C 13 T 3750,-1750;
C 13 T 3750,-11750;
C 19 T 3750,-21750;
C 12 T 7750,-3750;
C 12 T 7750,-7750;
C 12 T 7750,-13750;
C 12 T 7750,-17750;
C 13 T 7750,-1750;
C 13 T 7750,-11750;
C 19 T 7750,-21750;
C 12 T 11750,-3750;
C 12 T 11750,-7750;
C 12 T 11750,-13750;
C 12 T 11750,-17750;
C 13 T 11750,-1750;
C 13 T 11750,-11750;
C 20 T 11750,-21750;
C 23 T 14750,-20000;
C 12 T 15750,-3750;
C 12 T 15750,-7750;
C 12 T 15750,-13750;
C 12 T 15750,-17750;
C 13 T 15750,-1750;
C 13 T 15750,-11750;
C 20 T 15750,-21750;
C 24 T 17500,-18000;
C 16 T 19750,-3750;
C 16 T 19750,-7750;
```

```
C 15 T 19750,-13750;
C 15 T 19750,-17750;
C 16 T 19750,-1750;
C 16 T 19750,-11750;
C 28 T 19750,-21750;
C 12 R 0,-1 T 27000,-13750;
C 12 R 0,-1 T 27000,-17750;
C 12 R 0,-1 T 27000,-3750;
C 12 R 0,-1 T 27000,-7750;
C 17 T 23000,-11750;
C 14 R 0,-1 T 27000,0;
C 26 T 23000,-21750;
C 21 T 23750,-16750;
C 21 T 23750,-12750;
C 22 T 25750,-19500;
C 22 T 25750,-9500;
C 12 R 0,-1 T 31000,-13750;
C 12 R 0,-1 T 31000,-17750;
C 12 R 0,-1 T 31000,-3750;
C 12 R 0,-1 T 31000,-7750;
C 17 T 27000,-11750;
C 14 R 0,-1 T 31000,0;
C 27 T 27000,-20750;
C 13 R 0,-1 T 33000,-3750;
C 13 R 0,-1 T 33000,-7750;
C 13 R 0,-1 T 33000,-13750;
C 13 R 0,-1 T 33000,-17750;
C 18 T 31000,-11750;
C 25 R 0,-1 T 33000,0;
C 29 T 31000,-21750;
C 12 R 0,-1 T 37000,-13750;
C 12 R 0,-1 T 37000,-17750;
C 12 R 0,-1 T 37000,-3750;
C 12 R 0,-1 T 37000,-7750;
C 17 T 33000,-11750;
C 14 R 0,-1 T 37000,0;
C 30 T 33000,-21750;
C 12 R 0,-1 T 41000,-13750;
C 12 R 0,-1 T 41000,-17750;
C 12 R 0,-1 T 41000,-3750;
C 12 R 0,-1 T 41000,-7750;
C 17 T 37000,-11750;
C 14 R 0,-1 T 41000,0;
C 31 T 37000,-20750;
C 13 R 0,-1 T 43000,-3750;
C 13 R 0,-1 T 43000,-7750;
C 13 R 0,-1 T 43000,-13750;
C 13 R 0,-1 T 43000,-17750;
C 18 T 41000,-11750;
C 29 T 41000,-21750;
L NX;  B 500 500 42750,-36000;
DF;


DS 33;
9 SuperBuffer;
L NM;  B 5000 1000 2500,-7500;
       B 5000 1500 2500,-3500;
       B 5000 1500 2500,-750;
L ND;  B 1000 2000 500,-3250;
       B 1000 1500 500,-750;
       B 500 11000 500,-8500;
L NC;  B 500 1000 500,-3500;
       B 500 1000 500,-750;
L ND;  B 1500 2000 1250,-6250;
       B 1500 1000 1250,-9500;
L NP;  B 500 3250 1250,-6125;
       B 1500 500 1750,-7750;
       B 500 2750 1250,-9875;
L NM;  B 3000 750 2500,-11875;
       B 3000 750 2500,-13625;
L NP;  B 500 4750 1500,-2625;
L ND;  B 750 750 2125,-5625;
       B 1000 500 2250,-9750;
L NP;  B 500 1250 2250,-8125;
L NI;  B 1500 1500 3000,-6500;
L NP;  B 1500 500 3000,-6500;
L NI;  B 1500 2000 3000,-8750;
L ND;  B 500 4000 2500,-12000;
       B 500 2750 2500,-1375;
L NP;  B 1500 1000 3000,-8750;
L NM;  B 1500 1000 3250,-5500;
L ND;  B 1000 1000 3000,-5500;
L NM;  B 1500 1000 3250,-10000;
L ND;  B 1000 1000 3000,-10000;
L NC;  B 500 500 3000,-5500;
L ND;  B 500 4000 3000,-7750;
       B 1000 1000 3250,-7500;
L NC;  B 500 500 3000,-10000;
       B 500 500 3250,-7500;
L NP;  B 750 1000 3625,-5500;
L NC;  B 500 500 3500,-5500;
L NP;  B 500 5000 3500,-2750;
       B 750 1000 3625,-10000;
L NC;  B 500 500 3500,-10000;
L NP;  B 500 1000 3750,-6250;
       B 500 1250 3750,-8875;
L ND;  B 1000 1500 4500,-750;
       B 1000 2000 4500,-3250;
L NC;  B 500 1000 4500,-3500;
       B 500 1000 4500,-750;
DF;


DS 34;
9 InvertingSB;
C 33;
L NI;  B 1500 1500 1500,-1000;
L ND;  B 1750 500 1625,-1000;
       B 1500 2000 3500,-3250;
DF;


DS 35;
9 InvertingSBPair;
C 34 M X T 5000,0;
C 34 T 4000,0;
DF;


DS 36;
9 NoninvertingSB;
C 33;
L ND;  B 1500 2000 1500,-3250;
       B 1750 500 3375,-1000;
L NI;  B 1500 1500 3500,-1000;
DF;


DS 37;
9 NoninvertingSBPair;
C 36 M X T 5000,0;
C 36 T 4000,0;
DF;


DS 38;
9 SBExample;
C 35;
L NM;  B 13000 750 6500,-11875;
       B 13000 750 6500,-13625;
L NP;  B 1000 1000 3500,-11750;
L NM;  B 1000 1000 3500,-11750;
L NC;  B 500 500 3500,-11750;
L NP;  B 1000 1000 5500,-13500;
       B 500 2000 5250,-12250;
L NM;  B 1000 1000 5500,-13500;
L NC;  B 500 500 5500,-13500;
C 37 T 8000,0;
L ND;  B 1500 1000 11500,-10500;
       B 1500 1000 11500,-13750;
L NP;  B 1000 1000 11500,-11750;
       B 1000 1000 11500,-11750;
L NM;  B 1000 1000 11500,-11750;
L NC;  B 500 500 11500,-11750;
L NP;  B 500 3000 11500,-14250;
```

```
L ND;  B 1500 2000 13500,-14250;
L NP;  B 1000 1000 13500,-11750;
L NM;  B 1000 1000 13500,-11750;
L NC;  B 500 500 13500,-11750;
L NP;  B 500 3000 13500,-14250;
C 36 T 16000,0;
       B 500 4500 17250,-13500;
DF;


DS 39;
9 ShiftCell;
L NM;  B 1000 5000 500,-2500;
       B 1000 1000 500,-1750;
L ND;  B 1000 1000 500,-1750;
L NC;  B 500 500 500,-1750;
L ND;  B 3000 500 2000,-2000;
L NI;  B 3250 1500 2375,-2000;
L NP;  B 2250 1500 2375,-2000;
L NM;  B 1500 1000 3500,-2000;
L NC;  B 1000 500 3500,-2000;
L ND;  B 1000 1000 3750,-2000;
       B 500 1500 4000,-3000;
       B 5000 500 6500,-3500;
       B 2250 1000 5125,-2000;
L NP;  B 500 2250 4750,-1875;
       B 4500 500 7000,-1000;
L NM;  B 1000 5000 5750,-2500;
L NC;  B 500 500 5750,-2000;
L NP;  B 1000 1000 7500,-2500;
L NM;  B 1000 1000 7500,-2500;
       B 750 5000 7625,-2500;
L NC;  B 500 500 7500,-2500;
L NP;  B 500 1500 7500,-3500;
L NM;  B 1500 1000 9500,-1250;
       B 1500 1000 9500,-3250;
L ND;  B 1000 1000 9250,-3250;
L NP;  B 750 1000 9125,-1250;
L NC;  B 1000 500 9500,-1250;
       B 1000 500 9500,-3250;
L ND;  B 1000 1000 9750,-1250;
L NP;  B 750 1000 9875,-3250;
       B 2750 500 11125,-3000;
L ND;  B 5000 500 12750,-1500;
L NP;  B 1000 1000 11500,-500;
L NM;  B 1000 1000 11500,-500;
       B 750 5000 11375,-2500;
L NC;  B 500 500 11500,-500;
L NP;  B 500 1500 11500,-1500;
       B 2250 500 13125,-3000;
L NM;  B 1000 5000 13250,-2500;
L ND;  B 2250 1000 13875,-4000;
L NC;  B 500 500 13250,-4000;
L NP;  B 500 2250 14250,-3875;
L ND;  B 500 750 15000,-4625;
       B 500 1500 15000,-750;
L NM;  B 1500 1000 15500,-4000;
L ND;  B 1000 1000 15250,-4000;
L NI;  B 3250 1500 16625,-4000;
L NC;  B 1000 500 15500,-4000;
L ND;  B 3000 500 17000,-4000;
L NP;  B 2250 1500 16625,-4000;
L NM;  B 1000 5000 18500,-2500;
       B 1000 1000 18500,-3750;
L ND;  B 1000 1000 18500,-3750;
L NC;  B 500 500 18500,-3750;
DF;


DS 40;
9 PSCell;
L ND;  B 4000 500 2000,-1500;
L NP;  B 1000 1000 500,-2500;
L NM;  B 1000 1000 500,-2500;
       B 750 5000 375,-2500;
L NC;  B 500 500 500,-2500;
L NP;  B 500 1750 500,-1625;
C 39 M X T 20750,0;
L NM;  B 1000 1500 4250,-2000;
L NP;  B 1000 750 4250,-2375;
L ND;  B 1000 1000 4250,-1750;
L NP;  B 2750 500 5375,-2500;
L NC;  B 500 1000 4250,-2000;
DF;

( End of Library79-250.cif. );
( Demo design added October 23, 1979 );

DS 41;
9 Demo;
L ND;  B 2750 500 1375,-53250;
L NM;  B 1500 50250 1250,-43875;
       B 42750 750 21875,-72875;
       B 42750 750 21875,-74625;
L NP;  B 500 500 1750,-73500;
       B 1500 500 2250,-75500;
       B 1250 500 2375,-49750;
L NM;  B 2000 1000 3000,-38750;
       B 1750 1000 2875,-56750;
       B 750 1000 2375,-29250;
       B 5500 1500 4750,-61750;
       B 5500 1000 4750,-68500;
       B 1000 1000 2500,-19250;
L NP;  B 500 2000 2250,-74250;
       B 1500 500 2750,-54000;
       B 1500 500 2750,-56250;
L ND;  B 500 5000 2500,-50500;
L NM;  B 1000 1500 3000,-47750;
L ND;  B 1000 1000 3000,-48000;
L NP;  B 1000 750 3000,-47375;
       B 500 3750 2750,-45125;
       B 2250 500 3625,-43000;
       B 500 1750 2750,-55125;
C 20 T 2750,-23750;
L NC;  B 500 1000 3000,-47750;
L NP;  B 500 1250 3000,-65875;
C 14 T 3000,-15750;
       B 500 2750 3250,-22375;
       B 2000 500 4000,-20750;
L NI;  B 1500 3250 3750,-40625;
L NP;  B 1500 2250 3750,-40625;
L NM;  B 1000 1000 3500,-38750;
L ND;  B 1000 1000 3500,-38750;
L NP;  B 1000 1000 3500,-49750;
L NM;  B 1000 1000 3500,-49750;
C 14 T 3000,-11750;
C 14 T 3000,-3750;
       B 1000 2750 3500,-2375;
       B 19000 1000 12500,-500;
C 14 T 3000,-7750;
L NP;  B 750 500 3375,-73500;
       B 500 1000 3250,-66500;
L ND;  B 1000 3000 3750,-43000;
L NM;  B 1000 1500 3750,-41750;
L NC;  B 500 500 3500,-38750;
L NM;  B 1000 1000 3750,-44000;
L NC;  B 500 500 3500,-49750;
L NP;  B 500 500 3500,-74000;
       B 500 1000 3500,-67000;
L NC;  B 500 500 3750,-44000;
       B 500 1000 3750,-41750;
L ND;  B 500 2500 3750,-40500;
L NP;  B 500 1000 3750,-67500;
C 39 R 0,1 T 3750,-57250;
       B 500 1000 4000,-67000;
       B 500 1000 4250,-66500;
       B 500 1250 4500,-65875;
L ND;  B 500 500 4500,-42250;
L NP;  B 500 750 4750,-73125;
       B 1750 500 5375,-73750;
       B 1250 500 5375,-72750;
       B 1250 500 5375,-71750;
L NM;  B 1750 1000 5875,-22500;
       B 1750 1000 5875,-20500;
L NP;  B 1000 1000 5500,-20500;
       B 1000 1000 5500,-22500;
```

```
        B 500 750 5250,-23375;
L ND;   B 500 3750 5250,-40125;
L NC;   B 500 500 5500,-20500;
        B 500 500 5500,-22500;
L NP;   B 500 1000 6000,-72250;
C 23 T 5750,-18000;
C 23 T 5750,-14000;
C 23 T 5750,-16000;
C 23 T 5750,-12000;
        B 750 500 6875,-71750;
C 12 T 6750,-15750;
C 12 T 6750,-19750;
C 20 T 6750,-23750;
C 12 T 6750,-11750;
C 12 T 6750,-3750;
C 13 T 6750,-1750;
C 12 T 6750,-7750;
        B 500 500 7000,-72250;
C 35 T 7500,-61000;
C 39 R 0,1 T 7750,-57250;
C 24 T 8500,-10000;
C 24 T 8500,-8000;
C 24 T 8500,-6000;
C 24 T 8500,-4000;
L ND;   B 500 3750 9250,-40125;
L NM;   B 2250 1000 10875,-59750;
L ND;   B 1000 1000 10250,-59750;
        B 500 750 10000,-60625;
C 23 T 9750,-18000;
C 23 T 9750,-16000;
C 23 T 9750,-10000;
C 23 T 9750,-8000;
L NC;   B 500 500 10250,-59750;
L NP;   B 1000 1000 11000,-74500;
L NM;   B 1000 1000 11000,-74500;
C 12 T 10750,-15750;
C 12 T 10750,-19750;
C 20 T 10750,-23750;
C 12 T 10750,-11750;
C 12 T 10750,-3750;
C 13 T 10750,-1750;
C 12 T 10750,-7750;
L NC;   B 500 500 11000,-74500;
L NP;   B 500 1750 11250,-73125;
        B 1000 1000 11500,-59750;
L NC;   B 500 500 11500,-59750;
L NP;   B 500 1750 11750,-58375;
C 39 R 0,1 T 11750,-57250;
        B 4000 500 14000,-57750;
        B 1000 1000 13000,-72750;
L NM;   B 1000 1000 13000,-72750;
C 24 T 12500,-14000;
C 24 T 12500,-12000;
C 24 T 12500,-6000;
C 24 T 12500,-4000;
L NC;   B 500 500 13000,-72750;
L ND;   B 500 3750 13250,-40125;
L NM;   B 2250 1000 14875,-59750;
L ND;   B 1000 1000 14250,-59750;
        B 500 750 14000,-60625;
C 23 T 13750,-18000;
C 23 T 13750,-14000;
C 23 T 13750,-10000;
C 23 T 13750,-6000;
L NC;   B 500 500 14250,-59750;
C 12 T 14750,-15750;
C 12 T 14750,-19750;
C 20 T 14750,-23750;
C 12 T 14750,-11750;
C 12 T 14750,-3750;
C 13 T 14750,-1750;
C 12 T 14750,-7750;
L NP;   B 1000 1000 15500,-59750;
L NC;   B 500 500 15500,-59750;
C 35 T 15500,-61000;
L NP;   B 500 750 15750,-58875;
        B 1000 1000 16250,-49750;
L NM;   B 1000 1000 16250,-49750;
L NP;   B 500 7750 16250,-54125;
L NC;   B 500 500 16250,-49750;
L NP;   B 1500 500 16750,-58750;
C 24 T 16500,-16000;
C 24 T 16500,-12000;
C 24 T 16500,-8000;
C 24 T 16500,-4000;
L ND;   B 750 500 17125,-42250;
L NM;   B 7250 1000 20375,-56750;
        B 7250 1000 20375,-51500;
        B 7250 1000 20375,-44000;
        B 7250 1000 20375,-38750;
        B 3750 750 18625,-49625;
        B 500 750 17000,-45875;
L ND;   B 500 3750 17250,-40125;
L NP;   B 1000 1000 17750,-45750;
L NM;   B 1000 1000 17750,-45750;
        B 2250 1000 18375,-59750;
L ND;   B 1000 1000 17750,-59750;
L NP;   B 500 12750 17750,-52625;
L NC;   B 500 500 17750,-45750;
        B 500 500 17750,-59750;
L ND;   B 500 750 18000,-60625;
        B 2000 1000 19000,-74750;
L NP;   B 1000 1000 19000,-72750;
L NM;   B 1000 1000 19000,-72750;
L NP;   B 500 3750 18750,-75625;
        B 1000 1000 19000,-59750;
C 15 T 18750,-15750;
C 15 T 18750,-19750;
C 28 T 18750,-23750;
        B 750 500 19125,-37750;
C 15 T 18750,-11750;
C 15 T 18750,-3750;
C 16 T 18750,-1750;
C 15 T 18750,-7750;
L NC;   B 500 500 19000,-72750;
        B 500 500 19000,-59750;
L NM;   B 1000 3250 19500,-36625;
L NP;   B 500 21250 19250,-48625;
L NM;   B 750 3000 20125,-47750;
        B 4250 750 21875,-45875;
L ND;   B 500 1500 20000,-75750;
        B 2000 1000 21000,-76500;
L NP;   B 500 36250 20750,-41625;
        B 1000 1000 21000,-74500;
L NM;   B 1000 1000 21000,-74500;
L NP;   B 500 1750 20750,-73125;
L NM;   B 2250 1000 21625,-59750;
L NP;   B 1000 1000 21000,-59750;
L NC;   B 500 500 21000,-74500;
        B 500 500 21000,-59750;
L NP;   B 500 2000 21250,-76500;
        B 1000 500 21500,-24250;
        B 1000 1000 22250,-37000;
L NM;   B 1000 1000 22250,-37000;
L ND;   B 1000 1000 22250,-59750;
        B 500 750 22000,-60625;
        B 500 1500 22000,-75750;
C 12 R 0,-1 T 26000,-15750;
C 12 R 0,-1 T 26000,-19750;
L NP;   B 500 21500 22250,-48250;
L NC;   B 500 500 22250,-37000;
C 31 T 22000,-22750;
C 12 R 0,-1 T 26000,-11750;
C 12 R 0,-1 T 26000,-3750;
C 14 R 0,-1 T 26000,0;
C 12 R 0,-1 T 26000,-7750;
        B 500 500 22250,-59750;
L NP;   B 1500 500 23250,-58750;
L NM;   B 1250 750 23375,-36875;
C 21 T 22750,-18750;
L ND;   B 1000 1000 23500,-44000;
L NP;   B 1000 1000 23750,-49750;
L NM;   B 1000 1000 23750,-49750;
L NC;   B 500 500 23500,-44000;
L ND;   B 500 1000 23500,-43000;
        B 750 500 23625,-42250;
```

```
L NP;  B 500 7750 23750,-54125;
L NC;  B 500 500 23750,-49750;
C 35 T 23500,-61000;
C 40 M X R 0,1 T 24000,-36500;
L NP;  B 4000 500 26000,-57750;
L NM;  B 2250 1000 25125,-59750;
L NP;  B 1000 1000 24500,-59750;
       B 500 750 24250,-58875;
L NC;  B 500 500 24500,-59750;
C 22 T 24750,-21500;
L ND;  B 500 4250 25500,-34375;
       B 1000 1000 25750,-59750;
L NC;  B 500 500 25750,-59750;
L ND;  B 500 750 26000,-60625;
C 12 R 0,-1 T 30000,-15750;
C 12 R 0,-1 T 30000,-19750;
C 31 T 26000,-22750;
C 12 R 0,-1 T 30000,-11750;
C 12 R 0,-1 T 30000,-3750;
C 14 R 0,-1 T 30000,0;
C 12 R 0,-1 T 30000,-7750;
       B 2000 1000 27000,-74750;
L NP;  B 500 2750 26750,-75125;
       B 1000 1000 27000,-72750;
L NM;  B 1000 1000 27000,-72750;
L NP;  B 8250 500 30625,-76750;
C 21 T 26750,-18750;
L NC;  B 500 500 27000,-72750;
C 40 M X R 0,1 T 28000,-36500;
L ND;  B 2000 1000 29000,-74750;
L NM;  B 2250 1000 29125,-59750;
L NP;  B 1000 1000 28500,-59750;
       B 500 1750 28250,-58375;
L NC;  B 500 500 28500,-59750;
L NP;  B 1000 1000 29000,-72750;
L NM;  B 1000 1000 29000,-72750;
C 22 T 28750,-21500;
L NC;  B 500 500 29000,-72750;
L NP;  B 500 1750 29250,-74625;
       B 3500 500 30750,-75750;
L ND;  B 500 4250 29500,-34375;
       B 1000 1000 29750,-59750;
L NC;  B 500 500 29750,-59750;
L ND;  B 500 750 30000,-60625;
C 12 R 0,-1 T 34000,-15750;
C 12 R 0,-1 T 34000,-19750;
C 31 T 30000,-22750;
C 12 R 0,-1 T 34000,-11750;
C 12 R 0,-1 T 34000,-3750;
C 14 R 0,-1 T 34000,0;
C 12 R 0,-1 T 34000,-7750;
C 21 T 30750,-18750;
C 40 M X R 0,1 T 32000,-36500;
L NM;  B 9250 1500 37125,-64500;
L NP;  B 1750 1000 33375,-69750;
L NM;  B 1250 1000 33125,-68500;
       B 1500 1000 33250,-71000;
L NP;  B 750 1000 32875,-71000;
       B 500 5750 32750,-73125;
C 22 T 32750,-21500;
L ND;  B 1000 1000 33250,-68500;
L NI;  B 1500 2000 33500,-69750;
L NC;  B 1000 500 33250,-71000;
L ND;  B 2500 1000 34250,-71000;
L NC;  B 500 500 33250,-68500;
L ND;  B 500 4250 33500,-34375;
       B 500 2250 33500,-69625;
C 12 R 0,-1 T 38000,-15750;
C 12 R 0,-1 T 38000,-19750;
C 31 T 34000,-22750;
C 12 R 0,-1 T 38000,-11750;
C 12 R 0,-1 T 38000,-3750;
C 14 R 0,-1 T 38000,0;
C 12 R 0,-1 T 38000,-7750;
C 21 T 34750,-18750;
L NP;  B 500 7500 35000,-73750;
L ND;  B 1000 1000 35750,-64750;
       B 500 6500 35750,-68250;
L NC;  B 500 500 35750,-64750;
C 40 M X R 0,1 T 36000,-36500;
C 22 T 36750,-21500;
L ND;  B 500 4250 37500,-34375;
C 13 R 0,-1 T 40000,-15750;
C 13 R 0,-1 T 40000,-19750;
C 29 T 38000,-23750;
C 13 R 0,-1 T 40000,-11750;
C 13 R 0,-1 T 40000,-3750;
C 13 R 0,-1 T 40000,-7750;
L NP;  B 500 2250 39250,-62375;
       B 1500 500 40000,-61250;
       B 1500 500 40000,-63500;
L NM;  B 3250 1000 41625,-27500;
L NP;  B 1000 500 40500,-62500;
       B 3250 500 42125,-39750;
       B 500 500 40750,-61500;
       B 500 1000 40750,-63000;
L NM;  B 750 1000 41375,-51500;
       B 750 1000 41375,-44000;
L NP;  B 500 2250 41500,-37625;
       B 1500 500 42250,-38750;
       B 1500 500 42250,-36500;
L NM;  B 1500 37250 42500,-46625;
L NP;  B 500 2250 43000,-37625;
Df;

C 41;

End
```

Appendix E. Additional References

The following articles are not referenced directly in the main document but are included as background for those who are interested in pursuing individual topics further.

[Bell 1978]
A. T. Bell, "An Introduction to Plasma Processing", *Solid State Technology*, April, 1978.

[Beyerlein 1978]
F. W. Beyerlein, "New Developments in Automatic Wire Bonding Equipment", *Electronic Packaging and Production*, January, 1978.

[Blais 1977]
P. D. Blais, "Edge Acuity and Resolution in Positive Type Photoresist Systems", *Solid State Technology*, August, 1977.

[Cuthbert 1977]
J. D. Cuthbert, "Optical Projection Printing", *Solid State Technology*, August, 1977.

[Duchynski 1977]
R. J. Duchynski, "Ion Implantation for Semiconductor Devices", *Solid State Technology*, November, 1977.

[Frey 1978]
D. W. Frey and E. B. Hryhorenko, "Photoresists for Electronics", *Electronic Packaging and Production*, January, 1978.

[Glaser & Subak-Sharpe 1977]
A. B. Glaser and G. E. Subak-Sharpe, *Integrated Circuit Engineering*, Addison-Wesley, 1977.

[Gupta 1972]
A. Gupta and J. W. Lathrop, "Yield Analysis of Large Integrated-Circuit Chips", *IEEE Journal of Solid-State Circuits*, vol. sc-7, no. 5, October, 1972.

[Henriksen 1977]
G. M. Henriksen, "Automatic Photo-Composition of Reticles", *Solid State Technology*, August, 1977.

[Hughes 1977]
G. P. Hughes, "X-Ray Lithography for IC Processing", *Solid State Technology*, May, 1977.

[Keyes 1977]
R. W. Keyes, "Physical Limits in Semiconductor Electronics", *Science*, March, 1977.

[Levinthal 1977]
D. J. Levinthal, "Photoresist Trends in Semiconductor Processing", *Electronic Packaging and Production*, November, 1977.

[Markstein 1977]
H. W. Markstein, "Trends In Semiconductor Photolithography", *Electronic Packaging and Production*, March, 1977.

[Markstein 1978]
H. W. Markstein, "Wafer Sawing Update", *Electronic Packaging and Production*, March, 1978.

[Marshall 1974]
J. F. Marshall, "Scribing and Breaking of Semiconductor Wafers", *Solid State Technology*, September, 1974.

[McWilliams & Widdoes 1978]
T. M. McWilliams and L. C. Widdoes, Jr., "SCALD: Structured Computer-Aided Logic Design", *Proceedings of the 15th Annual Design Automation Conference*, June, 1978.

[Meindl 1977]
J. D. Meindl, "Microelectronic Circuit Elements", *Scientific American*, Sept., 1977.

[Oldham 1977]
W. G. Oldham, "The Fabrication of Microelectronic Circuits", *Scientific American*, Sept., 1977.

[O'Malley 1975]
A. J. O'Malley, "Technological Implications in the Photomasking Process", *Solid State Technology*, June, 1975.

[Roussel 1978]
J. Roussel, "Step-and-Repeat Wafer Imaging", *Solid State Technology*, May, 1978.

[Stapper 1976]
C. H. Stapper, "LSI Yield Modeling and Process Monitoring", *IBM Journal of Research and Development*, May, 1976.

XEROX

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304